Mike Faust

CREATE COOL SOLUTIONS

Starke Tools, Tipps & Tricks für Screens, Datenbank- und Web-Integration

1. Ausgabe, Juli 2002

Wir haben alle Anstrengungen unternommen, die Richtigkeit des in diesem Buch vorgestellten Materials zu gewährleisten. Es obliegt dem Leser, zu entscheiden, ob die Befehle, Programme und Informationen für den Einsatz in seiner Umgebung geeignet sind.

Kommentare sowie Verbesserungsvorschläge zu diesem Buch sind uns selbstverständlich stets willkommen.

Wir möchten Sie um Verständnis dafür bitten, daß für den Inhalt des Buchs keinerlei Garantie übernommen werden kann.

Copyright 2002 © I.T.P.-Verlags GMBH, Kaufering, Deutschland

Übersetzung: Dr. h. c. div. (ulc, usa) Stephan D. Petersen, 65529 Waldems
Telefon (+49) 06126 / 22 76 90
http://www.global-petersen.de

Satz: Verlagsdienstleistungen TextLift GbR, 86159 Augsburg
Telefon (+49) 0821 / 58 57 87

Übersetzte Orginalausgabe von Midrange Computing:
The
iSeries and AS/400
Programmer`s Guide to
Cool Things

Die teilweise oder vollständige Reproduktion dieses Materials ist untersagt.

Alle Rechte vorbehalten.

IBM, iSeries, AS/400 und OS/400 sind eingetragene Warenzeichen von IBM®.

ISBN: 3-930176-49-1

I.T.P.-Verlags-GmbH
Kolpingstraße 26
D-86916 Kaufering
Telefon (+49) 08191 / 96 49-0
Fax (+49) 08191 / 7 06 61
e-Mail: service@midrangemagazin.de
http://www.MidrangeMagazin.de

Dieses Buch ist meinem Vater Raymond (Jim) Faust gewidmet, der mir immer gesagt hat, dass alles Vorstellbare auch möglich ist.

Teil I
Cooles für Green Screens

Trotz der Dominanz der grafischen Windows-Anwendungen und der Web-Browser werden nach wie vor die meisten Routineaufgaben, die auf der AS/400 anfallen, mit Hilfe eines Dumb Terminals oder eines Terminalemulators erledigt. Aus diesem Grund stehen im Mittelpunkt dieses Abschnitts Aufgaben, die Sie eventuell noch nicht kennen und die von einem „Green Screen“ aus erledigt werden können. Wir beginnen mit der Vorstellung der Leistungsfähigkeit des DB2 Query Managers for AS/400. Wir werden Ihnen ein Dienstprogramm vorstellen, das es Ihnen erlaubt, problemlos Berichte zu erstellen, ohne eine einzige Zeile RPG-Code schreiben zu müssen und ohne Einbußen an Flexibilität – wie bei der Verwendung von Query/400 – hinnehmen zu müssen. Nachdem Sie festgestellt haben, wie leistungsfähig der Query Manager ist, werden Sie nie mehr ein Berichtsprogramm in RPG schreiben wollen.

Natürlich benötigen Sie nach wie vor RPG, um interaktive Anwendungen zu erstellen. Deshalb erfahren Sie in Kapitel 2, wie Sie Ihre Green-Screen-Anwendungen attraktiver gestalten können, indem Sie erweiterte Anzeigefunktionen verwenden. Die vorgestellte Gruppe von Display-File-Funktionen ermöglicht es Ihnen, den Green-Screen-Anwendungen das „Look and Feel“ eines Programms mit einer grafischen Benutzerschnittstelle (GUI) zu verleihen. Sie können somit in Green-Screen-Anzeigen Fenster, Schaltflächen, Kontrollkästchen, Optionsfelder und sogar ein Listenfeld erstellen. Sie können Ihre Anwendungen verschönern und gleichzeitig die Benutzerfreundlichkeit erhöhen, indem Sie nur wenige DDS-Schlüsselwörter hinzufügen.

Nachdem Sie die erweiterten Anzeigefunktionen kennen gelernt haben, sehen wir uns in Kapitel 3 an, wie in ILE RPG benutzerdefinierte Funktionen erstellt werden. Benutzerdefinierte Funktionen können verwendet werden, um Aufgaben von einfachen arithmetischen Berechnungen bis hin zu komplexen Funktionen auszuführen, mit denen die Daten physischer Dateien eingelesen und angezeigt werden. Anschließend wird ein ausgewählter Wert zurückgegeben. Sie werden eine Handvoll nützlicher Beispiele kennen lernen, die Sie in Ihre Anwendungen integrieren können. In diesem Kapitel erfahren Sie auch, wie FTP verwendet wird, um mehr zu tun, als nur Dateien zu senden und zu empfangen.

Sobald Sie diesen Abschnitt bearbeitet haben, werden Sie Ihren Programmier-Werkzeugkasten mit einer Fülle von neuen Werkzeugen ergänzt haben.

Den Code für alle Beispiele in diesem Buch finden Sie auf der beiligenden CD im Verzeichnis Codes.

I

Erstellen von Berichten mit Query Manager/400

Ein I.T.P.-Fachbuch

Als Erstes werden wir untersuchen, wie der Query Manager/400 als Alternative für das Schreiben von RPG-Programmen eingesetzt werden kann oder wie mit Query/400 angepasste Berichte erstellt werden können. QM/400 gibt Ihnen die Fähigkeit, gedruckte Berichte mit nahezu der gesamten Flexibilität zu erstellen, die beim Schreiben eines Programms in RPG zur Verfügung steht, und das in etwas weniger Zeit, als für das Erstellen einer Abfrage mit Hilfe von Query/400 benötigt wird. Am besten daran ist, dass Sie eine Query/400-Abfragedefinition problemlos in QM/400 konvertieren können.

Bevor wir beginnen, werde ich einige Begriffe definieren: *Query Management Queries* sind standardmäßig ein Teil von OS/400. Diese werden mit Hilfe von Quelldateien erstellt, die eine SQL-Anweisung enthalten, mit der die Daten für Ihre Abfrage definiert werden, sowie mit einer separaten Quellenteildatei, die die Definition des Berichtslayouts enthält. Der Begriff *Query Manager,* der in diesem Kapitel verwendet wird, bezieht sich auf den DB2 UDB Query Manager for AS/400, der ein Teil des DB2 Query Manager/ SQL Development-Kits mit der Lizenzprogrammnummer 5769ST1 ist. Dieses Produkt ist eine benutzerfreundliche Schnittstelle für Query-Management-Abfragen und -Formulare.

Für die Erstellung eines Berichts mit QM/400 werden zwei wichtige Komponenten benötigt: Die Query Manager-Abfrage, die verwendet wird, um zu definieren, welche Daten in Ihrem Bericht erscheinen werden, und das *Query Manager Report Form*, mit dem das Aussehen des Berichts einschließlich der Spalten- und Seitenüberschriften und Level-Breaks definiert wird. Das bedeutet, dass Sie ein Query Manager-Berichtsformular erstellen können, das von einer oder mehreren Query Manager-Abfragen ver-

wendet werden kann. Dies kann Ihnen dabei helfen, Ihre Entwicklungszeit bei der Erstellung neuer Berichte stark zu reduzieren, da Berichte mit einheitlichem Layout das gleiche Query Manager-Berichtsformular, jedoch individuelle Query Manager-Abfragen für die Definition der Datensätze verwenden können.

Die Flexibilität in QM/400 wird teilweise über die Möglichkeit erreicht, Parameter in Query Manager-Abfragen und -Formulare einzufügen. Mit Hilfe dieser Parameter können Informationen – wie z.B. Berichtskriterien (z.B. Kundennummern oder Datumsbereiche) – in Ihren Bericht übergeben werden. Sie können diese Parameter außerdem verwenden, um Informationen an Ihr Query Manager Berichtsformular zu übergeben, wenn Sie die Überschriften und Fußzeilen drucken. Die Möglichkeit, Parameter oder Feldwerte in Überschriften einzufügen, ist ein großartiges Feature, da Sie so die eingefügten Werte direkt im Bericht zeigen können. Wir befassen uns mit diesem Thema ausführlich etwas später in diesem Kapitel. Beginnen wir damit, die verschiedenen Methoden für die Erstellung von Query Manager Abfragen zu untersuchen.

Berichte können mit QM/400 mit drei Methoden erstellt werden:

- Konvertieren einer Query/400 QUERY-Definition
- Verwendung des Befehls STRQM, um das QM/400 Hauptmenü anzuzeigen
- Erstellen einer Query Manager-Abfrage, Generierung einer Quellenteildatei und Kompilierung dieser Datei.

Welche Methode für Sie die beste ist, hängt von Ihrem Startpunkt ab. Wenn z.B. eine Abfragedefinition vorhanden ist, die dem Be-

richt ähnelt, den Sie mit QM/400 erstellen möchten, können Sie diese Abfragedefinition in eine Query Manager-Abfrage und in ein -Formular konvertieren. Wenn Sie dagegen den Bericht vollständig neu erstellen, werden Sie vermutlich das QM/400-Menü verwenden wollen. In den meisten Fällen empfiehlt es sich nicht, die dritte Option zu wählen, denn die Erstellung der Source kann langwierig sein und hat gegenüber der Verwendung des QM/400-Menüs keine Vorteile.

Beginnen wir damit, eine Query/400-Abfragedefinition in QM/400 zu konvertieren. In diesem Prozess wird die Abfrage in eine Query Manager-Abfrage und in ein Query Manager-Berichtsformular aufgeteilt. Zuvor müssen die Source-Dateien für die Aufnahme der Query Manager-Abfrage und der Formular-Quellenteildateien erstellt werden. Verwenden Sie den Befehl Create Source Physical File (CRTSRCPF) und erstellen Sie zwei Source-Dateien mit dem Namen QQMQRYSRC, die verwendet werden, um die Abfrage-Source zu speichern, sowie QQMFORMSRC, mit dem die Formular-Source gespeichert wird.

```
CRTSRCPF FILE(LIB/QQMQRYSRC)
CRTSRCPF FILE(LIB/QQMFORMSRC)
```

Diese Dateien sind erforderlich, da wir, um eine Query/400-Abfragedefinition in QM/400 zu konvertieren, zuerst die Source für die Abfrage und das Formular aus der Abfragedefinition erstellen müssen.

Um die Source für die Query Manager-Abfrage zu erstellen, verwenden wir den Befehl Retrieve Query Manager Query (RTVQMQRY), indem wir folgenden Befehl eingeben:

```
RTVQMQRY QMQRY(Library/Query-Namen) SRCFILE(Library/QQMQRYSRC)
         ALWQRYDFN(*YES)
```

Der Schlüsselparameter ist hier der Parameter „Allow Information in a Query Definition“ (ALWQRYDFN), mit dem der Befehl aktiviert wird, um die Source für die Query Manager-Abfrage aus einer Query/400-Abfragedefinition zu erstellen.

Um den zweiten Teil des Puzzles fertig zu stellen (das Query Manager-Berichtsformular), müssen wir den Befehl Retrieve Query Manager Report Form (RTVQMFORM) verwenden. Dieser Befehl weist große Ähnlichkeit mit RTVQMQRY auf und wird ausgeführt durch die Eingabe:

```
RTVQMFORM QMFORM(Library/Query-Namen) SRCFILE(Library/QQMFORMSRC)
ALWQRYDFN(*YES)
```

Sobald wir diese Source abgerufen haben, müssen die QM/400-Objekte erstellt werden, indem diese Quellenteildateien kompiliert werden. Wir erstellen die Query Manager-Abfrage mit Hilfe des Befehls Create Query Manager Query (CRTQMQRY), indem wir Folgendes eingeben:

```
CRTQMQRY QMQRY(Library/qmquery) SRCFILE(Library/QQMQRYSRC)
```

Zuletzt verwenden wir den Befehl Create Query Manager Report Form (CRTQMFORM), um die Formular-Source zu kompilieren, indem Folgendes eingegeben wird:

```
CRTQMFORM QMFORM(Library/qmform)  SRCFILE(Library/QQMFORMSRC)
```

Jetzt denken Sie vielleicht, dass das viel Arbeit ist, um eine Abfragedefinition in QM/400 zu konvertieren, aber wenn Sie sehen, welche Funktionen QM/400 bietet, werden Sie verstehen, dass das Resultat diese Arbeit rechtfertigt.

Query Manager-Abfragen

Um mit den QM/400-Objekten zu arbeiten, müssen Sie das QM/400-Hauptmenü anzeigen. Der Zugriff erfolgt, indem Sie den Befehl Start Query Manager/400 (STRQM) eingeben, der keine Parameter besitzt. Abbildung 1.1 zeigt das Query Manager/400-Hauptmenü.

```
                        DB2 Query Manager for AS/400
                                                          System:   S1234ABC
Select one of the following:

     1. Work with Query Manager queries
     2. Work with Query Manager report forms
     3. Work with Query Manager tables

    10. Work with Query Manager profiles









Selection
     ==>

F3=Exit   F12=Cancel   F22=QM Statement
(C) COPYRIGHT IBM CORP. 1982, 1999.
```

Abbildung 1.1: Verwenden Sie das Query Manager-Hauptmenü, um Abfragen und Berichte zu erstellen.

Option 1 in diesem Menü wird verwendet, um Query Manager-Abfragen zu bearbeiten. Option 2 erlaubt es Ihnen, mit Query Manager-Berichtsformularen zu arbeiten. Option 3 ermöglicht es Ihnen, Tabellen für die Verwendung mit QM/400 zu erstellen oder zu modifizieren. Option 10 in diesem Menü erlaubt es Ihnen, die Zugriffsebene zu definieren, über die die individuellen Benutzerprofile verfügen. Geben Sie Option 1 ein, um den Bildschirm Work with

Query Manager Queries anzuzeigen, der in Abbildung 1.2 gezeigt wird.

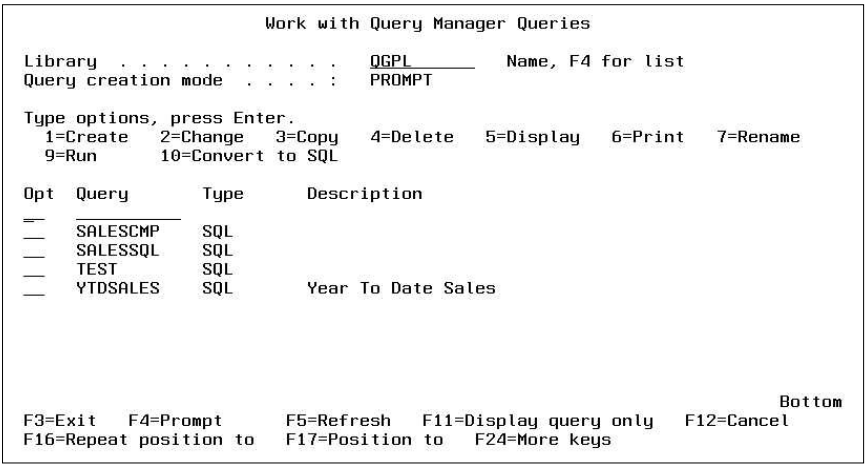


Abbildung 1.2: Verwenden Sie diese Anzeige, um Query Manager Queries zu erstellen und zu modifizieren.

Es gibt zwei unterschiedliche Abfragestile, die in diesem Bildschirm erstellt werden können: *bedienergeführt* und SQL. Eine bedienergeführte Abfrage wird unter Verwendung von ähnlichen Bildschirmen wie bei der Definition einer Abfrage in Query/400 erstellt. Wenn Sie im Bildschirm Define Prompted Query die Option Specify Tables wählen, können Sie auswählen, welche Datenbankdateien innerhalb Ihrer Abfrage verwendet werden und wie diese verknüpft werden. Der Bildschirm Define Expressions wird verwendet, um Felder in Ihrer Abfrage zu erstellen, deren Werte auf Grundlage der Felder in Ihrer Datenbankdatei berechnet werden, z.B. wenn ein Betrag auf Grundlage von Kosten/Einheit x-verkaufte Einheiten ermittelt wird. Die Felder Select und

Sequence Columns ermöglichen es Ihnen, auszuwählen, welche Felder aus Ihren Datenbankdateien in die Abfrage aufgenommen werden und in welcher Reihenfolge sie erscheinen sollen. Der Bildschirm Select Rows ermöglicht es Ihnen, anzugeben, welche Datensätze in Ihre Abfrage aufgenommen werden sollen. Select Summary Functions wird verwendet, wenn Datensätze gruppiert werden, um den Typ „Summe“ zu definieren (AVERAGE, SUM, COUNT etc.), der in den Spalten Ihrer Abfrage verwendet werden soll.

Die nächste Option im Bildschirm Define Prompted Query ist in Query/400 nicht vorhanden. Die Option Specify Duplicate Rows ermöglicht es Ihnen, dem Query Manager mitzuteilen, wie doppelte Datensätze innerhalb Ihrer Abfrage behandelt werden sollen – das heißt, Felder, die für Ihre Abfrage ausgewählt wurden, die identische Werte in zwei oder mehr Zeilen aufweisen. Die Optionen lauten:

1. Doppelt vorhandene Zeilen beibehalten.

2. Nur die erste Kopie jeder Zeile behalten.

Um geordnete Folgen für die Daten in Ihrer Abfrage zu erstellen, verwenden Sie den Bildschirm Define Sort Sequences. Der Bildschirm Define Report Formatting erlaubt es Ihnen, anzugeben, welches Query Manager-Berichtsformular mit dieser Abfrage verwendet werden soll. Im Allgemeinen sind bedienergeführte Abfragen für Personen geeignet, die mit SQL-Programmierung nicht vertraut sind oder die ihre ersten Schritte im Query Manager/400 wagen. Bedienergeführte Abfragen ermöglichen es Ihnen jedoch nicht, Parameter zu verwenden.

SQL-Abfragen hingegen werden unter Verwendung von SQL-Standardcode erstellt. Beispiel:

```
SELECT CUSNAM, CUSAD1, CUSAD2, CUSCTY, CUSSTE, CUSZIP
FROM CUSTMAST WHERE CUSSTE='PA'
```

Dieses Beispiel zeigt die Namens- und Adressfelder für alle Kunden in Pennsylvania an. Die Tatsache, dass dieser Abfragetyp die gesamte Leistungsfähigkeit von SQL verwendet und außerdem Parameter in Ihren SQL-Code eingeben kann, macht SQL-Abfragen viel flexibler als bedienergeführte Abfragen. Der Modus, den Sie wählen, um Ihre Abfragen zu erstellen, kann in Work with Query Manager Queries durch Drücken von F19 geändert werden. Der Standardmodus für jeden Benutzer wird im Query Manager-Profil konfiguriert, auf das über Option im Query Manager-Hauptmenü zugegriffen wird. Tabelle 1.1 zeigt eine Liste der einzelnen Funktionen im Bildschirm Query Manager Queries und Beschreibungen für ihre Verwendung.

Tabelle 1.1: Work with Query Manager Queries Optionen

Option #	Beschreibung
1	Erstellen einer neue Abfrage unter Verwendung des aktuellen Modus, wie ganz oben im Bildschirm gezeigt wird.
2	Ändern einer vorhandenen Abfrage. Der verwendete Modus basiert dabei auf dem Modus, der bei der letzten Speicherung der Abfrage genutzt wurde.
3	Erstellen einer Kopie der ausgewählten Abfrage.
4	Löschen einer vorhandenen Abfrage.
5	Anzeige der Abfragedefinition mit dem Modus, in dem die Abfrage zuletzt gespeichert wurde.
6	Ausdruck einer Kopie der SQL-Anweisung auf einem Drucker.
7	Umbenennen des ausgewählten QM-Abfrageobjekts.
8	Ausführen der ausgewählten Abfrage, nachdem Sie eine Reihe von Optionen für die Ausführung angezeigt haben.
9	Konvertieren einer Abfrage mit Bedienerführung in eine SQL-Abfrage.

Sie können im Bildschirm Work with Query Manager Queries eine bedienergeführte Abfrage in eine SQL-Abfrage konvertieren, indem Sie Option 10 verwenden. Beachten Sie dabei, dass dieser Prozess nicht umgekehrt werden kann und dass es keine Methode gibt, um eine SQL-Abfrage in eine bedienergeführte Abfrage zu konvertieren. Query Manager-Abfragen, die aus einer Query/400-Abfragedefinition erstellt werden, sind immer SQL-Abfragen.

Wenn Sie Parameter in Ihre SQL-Anweisungen einfügen, erhalten Sie die Möglichkeit, SQL-Abfragen zu erstellen, in die Sie Ihre Auswahlkriterien bei jeder Ausführung einspeisen. Das kann geschehen, indem Sie einfach eine Konstante in der WHERE-Klausel mit einem Variablennamen ersetzen, dem ein kommerzielles Pluszeichen (&) vorangestellt wird. Das Beispiel oben würde damit so aussehen:

```
SELECT CUSNAM, CUSAD1, CUSAD2, CUSCTY, CUSSTE, CUSZIP
    FROM CUSTMAST WHERE CUSSTE=&STATE
```

Wenn die Abfrage ausgeführt wird, können Sie den Wert für den Befehl Start Query Manager Query (STRQMQRY) bereitstellen, indem Sie den Parameter Set Variable (SETVAR) einsetzen. In unserem letzten Beispiel würde das so aussehen:

```
STRQMQRY CUSTLIST SETVAR(STATE 'PA')
```

Wenn kein SETVAR-Parameter angegeben wird, wird der Benutzer aufgefordert, den Wert einzugeben.

Wenn die verwendete Variable mit einem alphanumerischen Feld verglichen wird, muss der Wert, der in die Variable eingefügt wird, den erforderlichen Wert in einfachen Anführungszeichen enthalten.

Wenn keine Anführungszeichen verwendet werden, wird Query Manager den Wert als Feldnamen fehlinterpretieren – und ein Fehler wird zurückgegeben. Das bedeutet auch, dass Sie Feldnamen oder vollständige Bereiche des SQL-Codes als Variablen einfügen können. Sie können die gesamte oben gezeigte WHERE-Klausel mit einer Variable ersetzen und die WHERE-Klausel bei der Ausführung dynamisch erstellen. Damit können Sie generische Query Manager-Abfragen entwickeln und die Steuerung der Datenauswahl für Ihren Bericht einer CL-Anwendung überlassen.

In unserem Beispiel ersetzen wir daher die gesamte WHERE-Klausel mit der Variable &WHERE und verwenden danach den CL-Code in Abbildung 1.3. In diesem Programm wird, wenn nur ein Parameter eingegeben wird, die WHERE-Klausel nur für dieses Feld erstellt. Wenn mehrere Werte eingegeben werden, kombiniert sie das Programm innerhalb der WHERE-Klausel unter Verwendung von AND.

```
/*====================================================================*/
/* Kompilierung:                                                      */
/*                                                                    */
/* CRTCLPGM PGM(XXX/CLS001CL) SRCFILE(XXX/QCLSRC) */
/*                                                                    */
/*====================================================================*/
CLS001CL: PGM PARM(&STATE &ZIP &GROUP)

   DCL VAR(&STATE) TYPE(*CHAR) LEN(2)
   DCL VAR(&ZIP) TYPE(*CHAR) LEN(5)
   DCL VAR(&GROUP) TYPE(*CHAR) LEN(10)
   DCL VAR(&WHEREC) TYPE(*CHAR) LEN(128)

   IF COND(&STATE' ') THEN(DO)
   CHGVAR VAR(&WHEREC) VALUE('WHERE CUSSTE''' || &STATE || '''')
   ENDDO

   IF COND(&GROUP' ') THEN(DO)
   IF    COND(&WHEREC'') THEN(DO)
```

Abbildung 1.3: So erstellen Sie eine Query Management Where-Klausel in einem CL-Programm (Teil 1 von 2).

```
CHGVAR VAR(&WHEREC) VALUE(&WHEREC || '*AND CUSGRP''' || +
      &GROUP || ''') ELSE (DO)
CHGVAR VAR(&WHEREC) VALUE('WHERE CUSGRP''' || &GROUP || ''')
ENDDO
ENDDO

IF COND(&ZIP' ') THEN(DO)
IF    COND(&WHEREC'') THEN(DO)
CHGVAR VAR(&WHEREC) VALUE(&WHEREC || '*AND CUSZIP''' || +
      &ZIP || ''')
ELSE (DO)
  CHGVAR VAR(&WHEREC) VALUE('WHERE CUSZIP''' || &ZIP || ''')
  ENDDO
  ENDDO

  STRQMQRY QMQRY(CUSLST) SETVAR((WHEREC &WHEREC))

ENDPGM
```

Abbildung 1.3: So erstellen Sie eine Query Management Where-Klausel in einem CL-Programm in einem CL-Programm (Teil 2 von 2).

Zuletzt werden wir nach Erstellung der kompletten WHERE-Klausel die Abfrage mit dem Befehl STRQMQRY und dem Parameter SETVAR ausführen, wobei die Variable WHERE auf den Wert des Felds &WHEREC gesetzt werden wird. Wenn keine Parameter eingefügt werden, falls das Programm ausgeführt wird, lässt das Programm die WHERE-Klausel leer, was dazu führt, dass die QM-Abfrage alle Datensätze für die Datei anzeigt. Eine Standard-SQL-Anweisung hat folgendes Format:

```
SELECT CUSNAM FROM CUSTMAST WHERE CUSNO=1234
```

Zu den Konzepten der SQL-Sprache gehört es, dass der Zugriff auf die Datenbank über „natürlichsprachige“ Abfragen gewährt wird. So wird aus der Syntax sofort ersichtlich, dass z.B. im Beispiel oben das Feld Kundennamen („Customer Name“) aus der Kunden-Mastertabelle für alle Datensätze ausgewählt werden soll, deren Kundennummer gleich 1234 ist. Zusätzliche optionale Klauseln können z.B. die Klausel GROUP BY sein, die verwendet wird, um die Datensatzgruppierung zu definieren, und die Klausel ORDER BY, mit der festgelegt wird, wie die Datensätze sortiert werden. Obwohl wir insgesamt nicht viel Zeit mit der Besprechung von SQL-Funktionen verbringen werden, werden wir die Funktion CASE kurz näher erläutern.

Die Funktion CASE ermöglicht es Ihnen, den Wert eines Felds abhängig von den Bedingungen zu definieren. Sie können mit dieser leistungsfähigen Funktion bei berechneten Feldern Fehler vom Typ „Teilen durch Null“ (Divide by Zero) vermeiden. Hierzu geben Sie CASE – wie hier gezeigt – ein:

```
(CASE WHEN x = 0 THEN 0 ELSE y/x END)
```

In diesem Beispiel wird 0 als Ergebnis zurückgegeben, wenn *x* gleich 0 ist, anderenfalls wird das Ergebnis durch *x* geteilt. Der Operator CASE ist auch in anderen Situationen nützlich, z.B. wenn Sie Vertriebsdaten für eine angegebene Periode und das Jahr bis zum aktuellen Datum in Spalten zusammenfassen möchten.

Abbildung 1.4 zeigt den SQL-Code, der hierfür erforderlich ist:

```
SELECT ITEMNO, SLSPER, SLSYER, SUM(CASE WHEN SLSPER = &PERD
  THEN SLSAMT ELSE 0
END) AS PERD_SALES, SUM(CASE WHEN (SLSPER<= &PERD AND
  SLSYER = &YEAR) THEN
SLSAMT ELSE 0 END) AS YTD_SALES
FROM SALESFILE
WHERE SLSPER <= &PERD AND SLSYER = &YEAR
GROUP BY ITEMNO, SLSYER, SLSPER
ORDER BY ITEMNO, SLSYER, SLSPER
```

Abbildung 1.4: Dieser SQL-Code erstellt Spalten auf Grundlage von Datenwerten unter Verwendung der Funktion CASE

Auch hier verwenden wir wieder Parameter für die Definition der Periode und des Jahres für unsere Abfrage. Unser Feld für die ausgewählte Periode wird definiert, wenn die SUM- und CASE-Funktionen beide so verwendet werden, dass sie nur Daten für die benötigte Periode enthalten. Anschließend werden die Daten nach Positionsnummern zusammengefasst. Unser Feld YTD_Sales (Verkäufe während des Jahres bis heute) verwendet ebenfalls die Funktionen SUM und CASE, doch diesmal erfolgt die Auswahl aller Daten bis/einschließlich der gewählten Periode. Sie werden auch feststellen, dass wir die Variablen &PERD und &YEAR in der WHERE-Klausel angeben, die dafür sorgen, dass keine Datensätze unnötigerweise eingelesen werden, die von der Abfrage nicht benötigt werden.

So wird auch verhindert, dass die Abfrage Zeilen anzeigt, in denen sowohl die Werte für die aktuelle Periode als auch für YDT_Sales Null sind. Wenn wir ein Query Manager-Berichtsformular gemeinsam mit dieser Abfrage erstellen, können wir die Werte für SLSPER und SLSYER in die Seitenüberschriften einfügen. Ein Beispiel für den erforderlichen Bildschirm mit den Seitenüberschriften für das Query Manager-Berichtsformular, wird in Abbildung 1.5 gezeigt. Wir haben &2 verwendet, um die Periodennummer anzugeben, die sich in Spalte zwei unserer Abfrage befindet, sowie &3 für die Bezeichnung des Jahresfeldes in Spalte drei. Ein Beispiel für die Ausgabe sehen Sie in Abbildung 1.6.

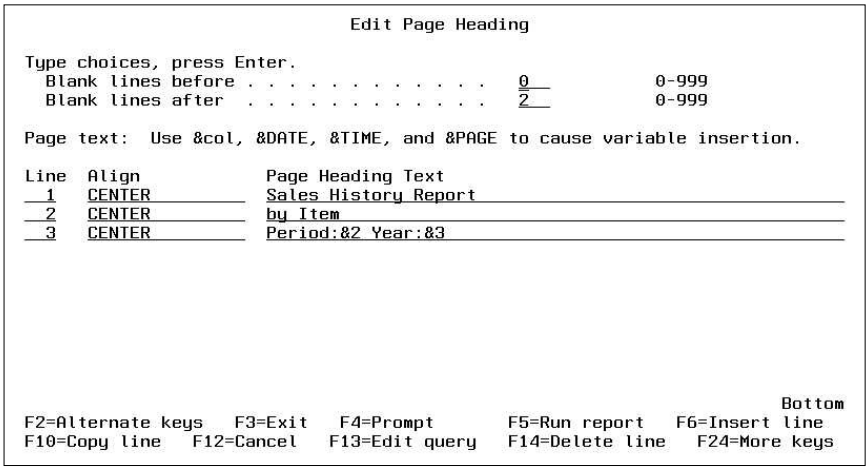
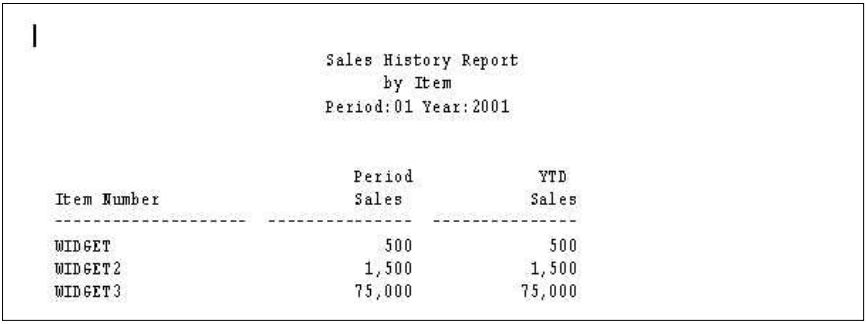


Abbildung 1.5: Werte aus einer Abfrage können in die Spaltenüberschrift eingefügt werden, wie hier gezeigt.



Sales History Report
by Item
Period:01 Year:2001

Item Number	Period Sales	YTD Sales
WIDGET	500	500
WIDGET2	1,500	1,500
WIDGET3	75,000	75,000

Abbildung 1.6: Wenn der Bericht gedruckt wird, werden die Werte in die Seitenüberschrift eingefügt.

Sie können in Ihre Query Manager-Abfrage auch Variablen einfügen, um diese in das Query Manager-Berichtsformular zu übergeben. Diese Technik kann sinnvoll sein, wenn Sie die Seitenüberschriften im Berichtsformular mit jobspezifischen Informationen ergänzen möchten, z.B. mit dem Unternehmensnamen, dem Benutzer oder der Workstation-ID. Das hier gezeigte Feld &USERID ist ein Beispiel für die Art und Weise, mit der dies mit Hilfe der Query Manager-Abfrage erfolgt:

```
SELECT CUSNO, CUSNAM, CUSAD1, CUSAD2, CUSCTY, CUSSTE, CUSZIP,
        &USERID AS USER FROM CUSTMAST;
```

Query Manager-Berichtsformulare

Wie bereits erklärt, wird mit dem Query Manager-Berichtsformular das *Wie* festgelegt, während die Query Manager-Abfrage verwendet wird, um zu definieren, *was* unser Bericht ist, d.h.: welche Daten erscheinen werden. Die unglaubliche Flexibilität, die mit QM/400 verfügbar wird, ist größtenteils auf die Funktionalität zurückzuführen, die mit Berichtsformularen verfügbar sind. Wenn Sie ein QM-Berichtsformular definieren, legen Sie normalerweise auch die Spaltenformatierung fest. Dies ist auch zufällig die erste Option, die in der in Abbildung 1.7 gezeigten Anzeige Select Report Format verfügbar ist. Dieser Bildschirm ermöglicht nicht nur die Definition der Größe und die Position der Spalten unseres Berichts, sondern auch die Definition der Art und Weise, wie ein Feld im Bericht verwendet wird.

```
                        Edit Column Formatting

Type information, press Enter.
For Usage and Edit, press F4 for list.
For Heading, press F4 for prompt.

Column          Heading                  Usage    Edit   Seq  Indent  Width
1               Item Number                                1       2     20
2               Period                   OMIT     I0       2       2      3
3               Year                     OMIT     I0       3       2      5
4               Period_Sales                      K0       4       2     15
5               YTD_Sales                         K0       5       2     15

                                                                      Bottom
F2=Alternate keys   F3=Exit   F4=Prompt   F5=Run report   F6=Insert line
F10=Copy line       F11=Edit heading      F12=Cancel      F24=More keys
```

Abbildung 1.7: Das Aussehen der gedruckten Spalten wird von diesem Bildschirm aus gesteuert.

Wenn zu diesem Zeitpunkt noch keine Abfrage aktiv ist, d.h.: Wenn Sie die Abfrage nicht im Fenster Query Manager-Abfrage bearbeiten, drücken Sie F13, um eine Abfrage für die Verwendung mit dem Bericht anzugeben. Sie können die Spalten in diese Anzeige von einer aktiven Abfrage laden, indem Sie F19 drücken. Sie können auch Spalten manuell hinzufügen, indem Sie F6 drücken.

USAGE legt fest, wie die Spalte im Bericht verwendet werden soll. Wenn USAGE leer ist, wird die Spalte einfach im Bericht ausgedruckt. Das Schlüsselwort OMIT unterdrückt den Ausdruck der Spalte. Die Optionen AVERAGE, SUM, COUNT, FIRST, LAST, MIN und MAX sind alle verfügbar und führen die ihrem Namen entsprechenden Funktionen (Durchschnitt, Summe, Anzahl, Erster, Letzter, Minimum, Maximum) aus. Das Schlüsselwort BREAKX mit Wert *x* ist ein Wert von 1 bis 6 und erlaubt Ihnen, bis zu sechs Ebenen mit Report Level-Breaks zu erstellen. Die Spalte Edit Codes wird ähnlich wie Edit-Codes in RPG oder in Query/400 verwendet. Die Ausnahme hier ist, dass Edit-Codes auch für nichtnumerische Felder festgelegt werden können, um die Art und Weise des Ausdrucks zu definieren. Eine Liste der gültigen Edit-Codes und ihre Verwendung wird in Tabelle 1.2 gezeigt.

Tabelle 1.2: Query Manager Report Form Edit Codes		
Daten-Typ	**Edit-Code**	**Beschreibung**
Character (Zeichen)	C	Daten über die angezeigte Breite abschneiden.
Character	CW	Umbruch bei angezeigter Feldbreite
Character	CT	Wortumbruch bei angezeigter Feldbreite
Numeric (Numerisch)	D*n*	Ausdruck mit Währungszeichen, Kommas und negativen Vorzeichen. Führende Nullen werden nicht angezeigt. *n*=Dezimalstellen Beispiel:-$12,345.67
Numeric	I*n*	Ausdruck mit führenden Nullen und negativen Vorzeichen. Keine Währungszeichen oder Kommas anzeigen. *n*=Dezimalstellen Beispiel:- -0012345.67
Numeric	J*n*	Ausdruck mit führenden Nullen, keine negativen Vorzeichen, Währungszeichen oder Kommas werden nicht angezeigt. n=Dezimalstellen Beispiel: 00012345.34
Numeric	K*n*	Ausdruck mit Währungszeichen, Kommas und negativen Vorzeichen. Führende Nullen werden nicht angezeigt. *n*=Dezimalstellen Beispiel: -12,345.67
Numeric	L*n*	Ausdruck mit negativen Vorzeichen. Keine Anzeige führender Nullen, Währungszeichen oder Kommas. n=Dezimalstellen Beispiel:-12345.67

Tabelle 1.2: Query Manager Report Form Edit Codes (Fortsetzung)		
Daten-Typ	**Edit-Code**	**Beschreibung**
Numeric	E*n*	Ausdruck im exponentiellen numerischen Format mit n-Dezimalstellen
Numeric	L*n*	Ausdruck mit Prozentzeichen, Kommas und negativen Vorzeichen. Führende Nullen werden nicht angezeigt. n=Dezimalstellen Beispiel: -12,345.67%
Datum/ Uhrzeit	TDY	Datum: Jahr/Monat/Tag mit 4-stelligem Jahr
Datum/ Uhrzeit	TDM	Datum: Monat/Tag/Jahr mit 4-stelligem Jahr
Datum/ Uhrzeit	TDD	Datum: Tag//Monat/Jahr mit 4-stelligem Jahr
Datum/ Uhrzeit	TDYA	Datum: Jahr/Monat/Tag mit 2-stelligem Jahr
Datum/ Uhrzeit	TDMA	Datum: Monat/Tag/Jahr mit 2-stelligem Jahr
Datum/ Uhrzeit	TDDA	Datum: Tag//Monat/Jahr mit 2-stelligem Jahr
Datum/ Uhrzeit	TTS	Uhrzeit: Stunden:Minuten:Sekunden. 24-Stunden-Format
Datum/ Uhrzeit	TTC	Uhrzeit: Stunden:Minuten:Sekunden. 12-Stunden-Format
Datum/ Uhrzeit	TTA	Uhrzeit: Stunden:Minuten. 24-Stunden-Format
Datum/ Uhrzeit	TTU	Uhrzeit: Stunden:Minuten:Sekunden. 12-Stunden-Format

Tabelle 1.2: Query Manager Report Form Edit Codes (Fortsetzung)		
Daten-Typ	**Edit-Code**	**Beschreibung**
Datum/ Uhrzeit	TTAN	Uhrzeit: Stunden:Minuten. 24-Stunden-Format, kein Trennzeichen (:)
Datum/ Uhrzeit	TSI	Uhrzeit: Zeitstempel (yyyy-mm-dd-hh-mm-ss-ms)
Graphische Daten	G	Daten über die definierte Spaltenlänge abkürzen
Graphische Daten	GW	An ausgewählter Spaltenbreite umbrechen.

Der Bildschirm Edit Page Headings wird verwendet, um die Seitenüberschriften für unseren Bericht festzulegen. Wenn wir die Anweisung SELECT verwenden würden, die den Parameter &USERID enthält, der in einem früheren Beispiel vorgestellt wurde, würden wir anschließend den Wert value &8 in die Seitenüberschrift auf unserem Berichtsformular einsetzen, um auf die achte Spalte unserer Abfrage zu verweisen (siehe Abbildung 1.8).

An diesem Beispiel wird Ihnen auffallen, dass die Seitenüberschriften in mehreren Zeilen definiert werden können. Sie können auch links, mittig und/oder rechts ausgerichtete Abschnitte der einzelnen Zeilen definieren (mit LEFT, CENTER und/oder RIGHT). Sie sehen, dass wir in Abbildung 1.8 die Systemvariable (&TIME) für die aktuelle Uhrzeit in Zeile 1 der Seitenüberschrift rechts ausrichten. Wir zeigen außerdem den Berichtstitel (CUSTOMER LISTING) in der Mitte der Seitenüberschrift an. Ebenfalls in dieser Zeile fügen wir links auf der Seite den Wert value &8 ebenfalls in Zeile 1 ein, wie bereits erklärt. Ganz oben im Bildschirm Edit Page Heading können Sie die Anzahl der Zeilen für den Zeilenvorschub vor und nach

dem Ausdruck der Seitenüberschrift als Zahl zwischen 0 und 999 Zeilen definieren. Die Fußzeilen der Seite werden auf die gleiche Weise definiert wie Überschriften, doch dies geschieht über den Bildschirm Edit Page Footings.

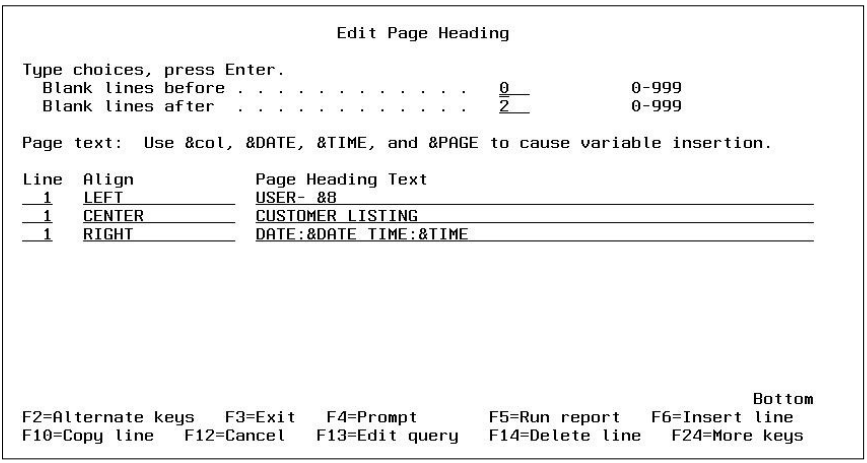


Abbildung 1.8: Seitenüberschriften können Werte von Spaltendaten enthalten.

Der in Abbildung 1.9 gezeigte Bildschirm Edit Final Text wird verwendet, um Text zu erstellen, der am Ende Ihres Berichts gedruckt wird. Obwohl diese Textzeilen festgelegt werden, indem ein ähnlicher Bildschirm wie bei der Festlegung der Seitenüberschriften und Fußzeilen verwendet wird, gibt es einige wichtige Unterschiede.

Als Erstes wird die Option New Page for Final Text verwendet, um festzulegen, ob der endgültige Text in eine eigene Seite am Ende des Berichts eingefügt werden soll.

Zweitens definiert die Option Second Blank Lines Before, wie viele Zeilen Zeilenvorschub zwischen der letzten Zeile des Berichts und dem abschließenden Text eingefügt werden. Diese Option kann entweder einen Wert zwischen 9 und 999 Zeilen enthalten oder den Sonderwert BOTTOM, der dazu führt, dass der abschließende Text ganz unten auf der Seite ausgedruckt wird.

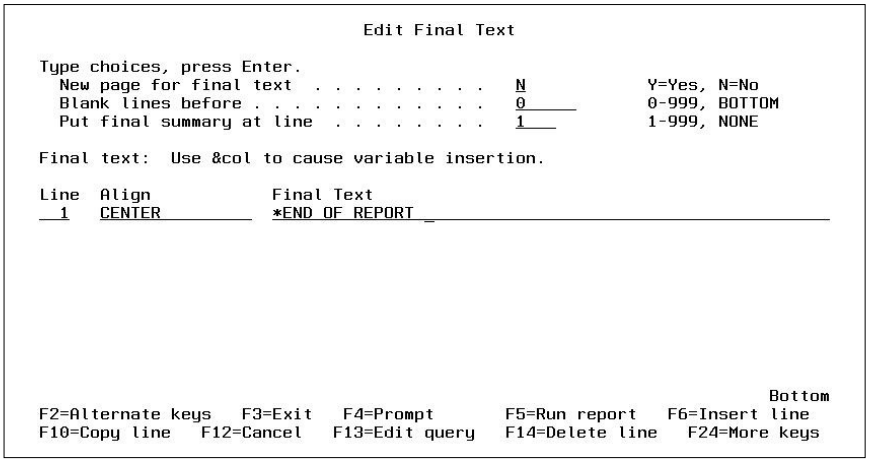


Abbildung 1.9: Der Bildschirm Edit Final Text wird verwendet, um Text für den Ausdruck am Ende unseres Berichts zu definieren.

Als Letztes wird die Option Put Final Summary at Line verwendet, um festzulegen, in welcher Zeile innerhalb der endgültigen Textübersicht die Daten angezeigt werden sollten. Dieser Parameter erhält entweder einen numerischen Wert zwischen 0 und der Anzahl der Zeilen im endgültigen Text oder den Sonderwert NONE, wenn in den endgültigen Text keine Übersicht eingefügt werden soll.

Der Bildschirm Specify Break Text, auf den vom Bildschirm Specify Report Formatting aus zugegriffen wird, wird verwendet, um Level-Breaks in Ihrem Bericht zu formatieren. Für jeden Level-Break in Ihrem Bericht kann sowohl Kopfzeilen- als auch Fußzeilentext definiert werden. Wenn Sie die Option 1 neben einem Break-Level in diesem Bildschirm angeben, wird der Bildschirm Edit Break Heading angezeigt. Dieser Bildschirm wird fast auf die gleiche Weise verwendet wie der Edit-Final-Text-Bildschirm. Sie können definieren, ob eine leere Seite vor dem Ausdruck der Break-Level-Überschrift eingeschoben werden soll, und Sie können wählen, ob die Spaltenüberschriften nach dem Ausdruck der Break-Level-Überschrift wiederholt werden sollen.

Sie können auch angeben, wie viele Zeilen vor und nach der Überschrift eingeschoben werden sollen. Dieser Wert wird ebenfalls als Zahl zwischen 0 und 999 angegeben und akzeptiert keine Sonderwerte. Wenn Sie eine Break-Level-Fußzeile angeben, indem Sie Option 2 im Bildschirm Specify Break Text wählen, erhalten Sie die Möglichkeit, die Break-Fußzeile auf eine neue Seite einzufügen. Wie bei Break-Headings können wir auch die Anzahl der Zeilen angeben, die vorher und nachher eingefügt werden sollen. Wir können aber auch den Sonderwert BOTTOM in der Option Blank Lines Before verwenden, damit unsere Break-Fußzeile ganz unten auf der Seite erscheint.

Zusätzlich wird die Option Put Break Summary at Line wie im Bildschirm Edit Final Text verwendet, um die Zeilenzahl innerhalb der Fußzeilen zu definieren, bei denen keine Übersichtsdaten angezeigt werden sollen. Diese Option akzeptiert einen Wert zwischen 0 und der Anzahl der Zeilen in den Fußzeilen zusätzlich zum Sonderwert NONE, der verwendet wird, um Übersichtsdaten aus den Fußzeilen auszulassen.

Die letzte Option im Bildschirm Select Report Format ist Specify Formatting Options. Dieser Bildschirm, den Sie in Abbildung 1.10 sehen, wird verwendet, um eine Reihe von allgemeinen Optionen für den Bericht zu definieren. Die Detailzeilenabstände können in diesem Wert als Zahl zwischen 1 und 4 festgelegt werden. Die Übersichtsansicht für Break-Spalten erlaubt es Ihnen, Ihren Bericht in einem Übersichtsformat anzuzeigen. In diesem Format werden Break-Werte bei der ersten Detailzeile jeder Gruppe angezeigt.

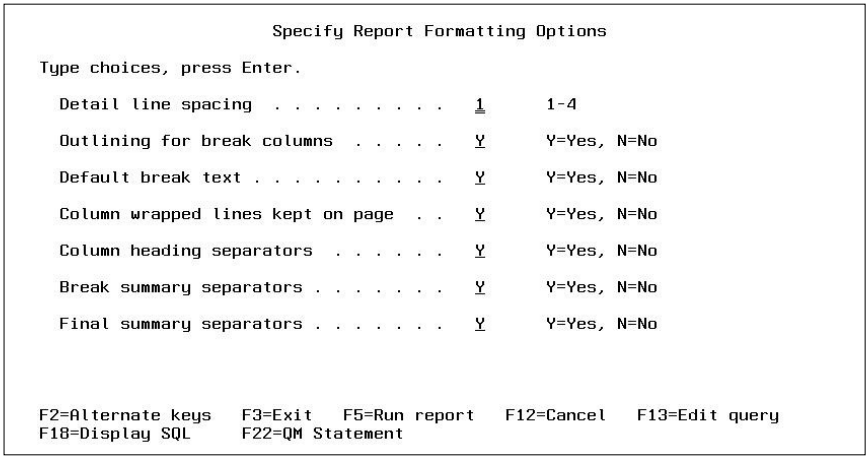


Abbildung 1.10: Der Bildschirm Specify Formatting Options wird verwendet, um allgemeine Berichtsoptionen zu definieren.

Die Option Default Break Text, die als Y oder N angegeben wird, sorgt dafür, dass eines oder mehrere Sternchen als Break-Fußzeilentext für alle Break-Levels angezeigt wird, für die kein Fußzeilentext definiert ist. Um festzulegen, ob umgebrochene Zeilen von Text in einem Textfeld im Bericht in Seitenumbrüchen zusam-

mengehalten werden sollten, verwenden Sie den Parameter Column-Wrapped Lines Kept on Page. Um festzulegen, ob eine Reihe von Bindestrichen zwischen Ihren Spaltenüberschriften und der ersten Zeile von Daten ausgedruckt werden sollten, öffnen Sie den Bildschirm Column Heading Separators.

Verwenden Sie die Optionen Break Summary Separators und Final Summary Separators, um anzugeben, ob Sie möchten, dass vor dem Ende dieser Abschnitte Trennzeichen ausgegeben werden sollen, bevor der Ausdruck erfolgt. Bei Break-Übersichten wird ein Bindestrichzeichen (-) verwendet. Bei Abschluss-Übersichten wird das Gleichheitszeichen (=) verwendet. Diese Funktionalität verhilft Ihnen zu der Flexibilität, die von den Query Manager-Berichtsformularen angeboten wird.

Als Nächstes bearbeiten wir Schritt für Schritt den Prozess der Erstellung eines einfachen Verkaufsberichts unter Verwendung des Query Manager. Als Erstes müssen unsere Verkaufszahlen unter Verwendung der DDS-Source in Abbildung 1.11 erstellt werden:

```
*=====================================================================*
* Kompilierung:                                                       *
*                                                                     *
* CRTPF FILE(XXX/SLS001PF) SRCFILE(XXX/QDDSSRC)                       *
*                                                                     *
*=====================================================================*
A            R SALES
A            SLSITM    15A
A            SLSCUS     8A
A            SLSPER     3 0
A            SLSYER     4 0
A            SLSQTY    15 2
A            SLSAMT    15 2
A            SLSCNM    30A
A            SLSDES    30A
```

Abbildung 1.11: SLS001PF Physische Datei für die Verwendung mit dem Query Manager-Beispiel.

Sobald Sie diese Datei erstellt haben, beginnen Sie mit der Erstellung der Query Manager-Abfrage mit Hilfe des Befehls Start Query Manager (STRQM). Wählen Sie Option 1 im Hauptmenü des Query Managers. Wenn ganz oben im Bildschirm als Query Creation Mode nicht SQL angezeigt wird, drücken Sie F19, um von der Bedienerführung zu SQL zu wechseln. Sie können später mit der gleichen Taste zurück in den Bedienerführungsmodus wechseln. Als Nächstes fügen Sie in die erste Zeile der Spalte Opt eine 1 ein, um eine neue Query Manager-Abfrage zu erstellen. Nun verwenden Sie den Code, der in Abbildung 1.12 gezeigt wird, als SQL für Ihre Abfrage.

```
SELECT SLSITM, SLSDES, SLSCUS, SLSCNM, SLSPER, SLSYER,
  SLSAMT, SLSQTY
FROM SLS001PF
WHERE SLSQTY0 AND SLSITM= &ITEM
ORDER BY SLSITM, SLSCUS
```

Abbildung 1.12: Dieser SQL-Code wird für unsere Query Manager Abfrage verwendet.

Wie Sie sehen, wird ein Parameter (&ITEM), oder eine *globale Variable,* wie dieser Parameter manchmal bezeichnet wird, verwendet, um unsere Positionsnummer auszuwählen. Nachdem Sie diese SQL-Befehle eingegeben haben, drücken Sie F15, um die Befehlssyntax Ihrer SQL-Anweisung zu prüfen. Wenn Fehler zurückgegeben werden, prüfen Sie erneut die SQL-Anweisung, um sicherzustellen, dass in der Source keine Tippfehler eingegeben wurden. Anderenfalls drücken Sie F3, um den Query Designer zu beenden, und speichern Sie die neue Abfrage unter dem Namen SLS001QMQ. Nun drücken Sie die Taste F12, um zum Query Manager-Hauptmenü zurückzukehren. Wählen Sie Option 2 für die Anzeige von Work with Query Manager Report Form. Wäh-

len Sie erneut Option 1 in der ersten Zeile unter der Spalte Opt. Wenn der Bildschirm in Abbildung 1.13 angezeigt wird, wählen Sie alle Edit Column Formatting-Optionen, die Option Edit Page Heading und die Option Edit Break Text. Drücken Sie dann die Eingabetaste.

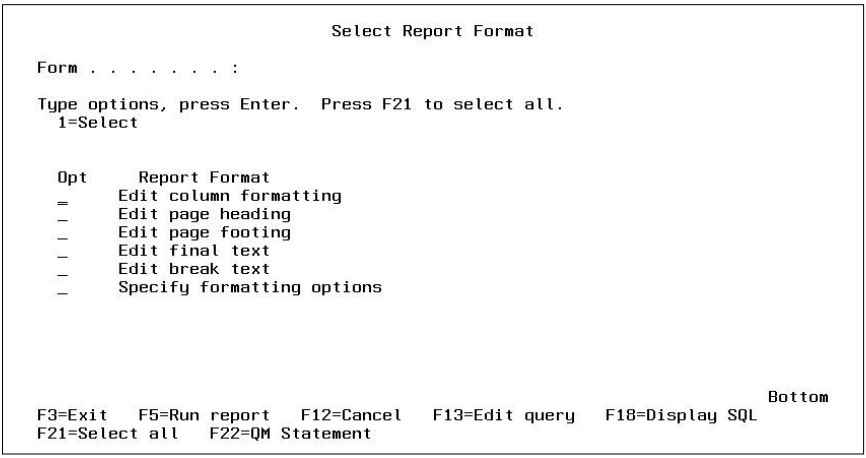


Abbildung 1.13: In diesem Bildschirm werden Query Manager Report Forms definiert.

Damit werden die einzelnen Bildschirme nacheinander angezeigt. Wenn der Bildschirm Edit Column Formatting angezeigt wird, sollten Sie die Spalten von der Query Manager-Abfrage sehen, die wir im vorherigen Schritt erstellt haben. Wenn das nicht der Fall ist, drücken Sie F5, um den Bericht anzuzeigen. Anschließend werden Sie dazu aufgefordert, den Namen der zu verwendenden Abfrage anzugeben. Geben Sie als Abfragenamen SLS001QMQ ein und geben Sie in das bereitgestellte Feld den Bibliotheksnamen ein.

Sie sollten nun aufgefordert werden, den &ITEM-Parameter einzugeben. Vergessen Sie bei der Eingabe nicht, diesen Wert in einfache Anführungszeichen zu setzen (, ,). Nachdem Sie diesen Wert eingegeben und die Eingabetaste gedrückt haben, erhalten Sie eventuell eine Nachricht, die Sie informiert, dass das Formular der Abfrage nicht entspricht. Der Grund dafür ist die Differenz bei der Anzahl der Spalten. Drücken Sie F19, um die Spalten aus der Abfrage zu laden. Sie sollten den in Abbildung 1.14 gezeigten Bildschirm sehen. Obwohl wir die Positionsnummer und die Beschreibungsfelder in unsere Abfrage aufgenommen haben, möchten wir diese im Bericht nicht als Spalten anzeigen.

Diese Spalten sind überzählig, weil der Bericht für eine angegebene Position ausgeführt werden wird. Wir werden sie jedoch in unsere Seitenüberschrift aufnehmen. Um diese Werte aus dem Ausdruck auf Spaltenebene auszuschließen, geben Sie für SLSITM und SLSDES in der Spalte USAGE den Wert OMIT ein.

```
                        Edit Column Formatting

Type information, press Enter.
For Usage and Edit, press F4 for list.
For Heading, press F4 for prompt.

Column          Heading                 Usage    Edit   Seq  Indent  Width
SLSITM          SLSITM                  OMIT            1      2      15
SLSDES          SLSDES                  OMIT            2      2      25
SLSCUS          Cust Num                BREAK1          3      2       8
SLSCNM          Customer Name                           4      2      20
SLSPER          Mth                             L0      5      2       3
SLSYER          Year                            L0      6      2       5
SLSAMT          Sales_Value                             7      2      15
SLSQTY          Units_Sold                              8      2      15




                                                                    Bottom
F2=Alternate keys   F3=Exit   F4=Prompt   F5=Display report   F6=Insert line
F10=Copy line       F11=Edit heading      F12=Cancel          F24=More keys
```

Abbildung 1.14: In diesem Bildschirm definieren wir das Aussehen unserer Spalten.

Nun müssen wir unsere Spaltenüberschriften so ändern, dass sie aussagekräftiger sind. Drücken Sie F11, um die Edit Column Headings-Ansicht des Bildschirms umzuschalten. Nun ersetzen Sie „SLSCUS“ mit „Cust Num“, „SLSCNM“ mit „Customer Name“, „SLSPER“ mit „Month“, „SLSYER“ mit „Year“, „SLSAMT“ mit „Sales_Value“ – und zuletzt „SLSQTY“ mit „Units_Sold“.

Verwenden Sie das Unterstrichzeichen, um dafür zu sorgen, dass der Query Manager den Spaltentext in eine neue Zeile einfügt. Drücken Sie die Eingabetaste, um den Bildschirm Edit Page Headings anzuzeigen. Den Bildschirm für unser Beispiel sehen Sie in Abbildung 1.15. Sie werden feststellen, dass wir in der zweiten Zeile die Spalten 1 und 2 in den Seitenüberschriften durch Verweis als &1 und &2 eingefügt haben.

Als Resultat ersetzen diese Parameter im ausgedruckten Bericht die Positionsnummer und die Beschreibung.

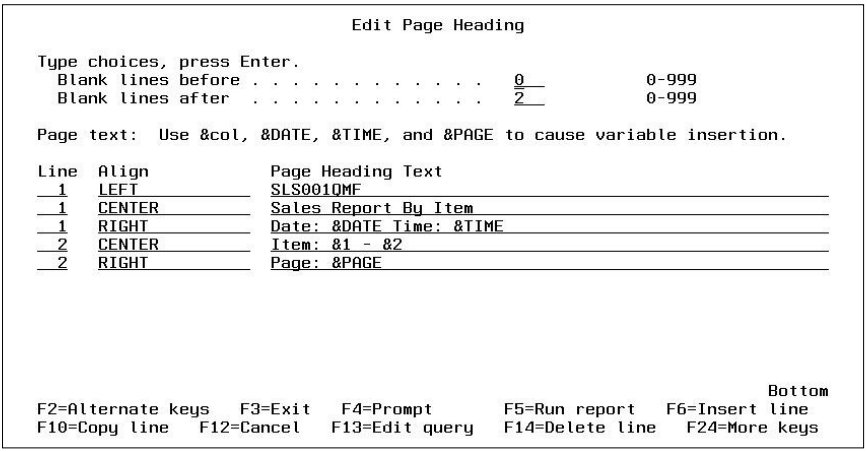


Abbildung 1.15: Wir verweisen auf die Werte der Spalten 1 und 2 unter Verwendung von &1 und &2.

Geben Sie die in Abbildung 1.15 gezeigten Informationen für Ihre Seitenüberschriften ein und drücken Sie die Eingabetaste.

Fügen Sie im Bildschirm Edit Break Text neben dem Break-Level für Feld SLSCUS eine 2 ein. Abbildung 1.16 zeigt den Edit-Break-Footing-Bildschirm für dieses Feld. Auch hier wird auf eine Spalte verwiesen, in diesem Fall auf die Kundennummer. Verwenden Sie diese Informationen, um Ihre Break-Fußzeilen zu bearbeiten. Drücken Sie die Eingabetaste, um zum Bildschirm Edit Report Format zurückzukehren.

Nun drücken Sie F5, um Ihren Bericht anzuzeigen. Da unsere Datei keine Daten enthält, werden keine Datensätze angezeigt, aber Sie werden die Spaltenüberschriften sehen. Drücken Sie F3, um das Query Manager-Berichtsformular zu speichern und Ihre Arbeit zu beenden. Verwenden Sie ein Dateidienstprogramm Ihrer Wahl, um Dummy-Daten in Ihre Datei einzugeben und führen Sie anschließend Ihren Bericht mit folgendem Befehl erneut aus:

```
STRQMQRY QMQRY(MFAUST/SLS001QMQ)
  OUTPUT(*PRINT)
  QMFORM(MFAUST/SLS001QMF)
  DATETIME(*NO) PAGNBR(*NO)
```

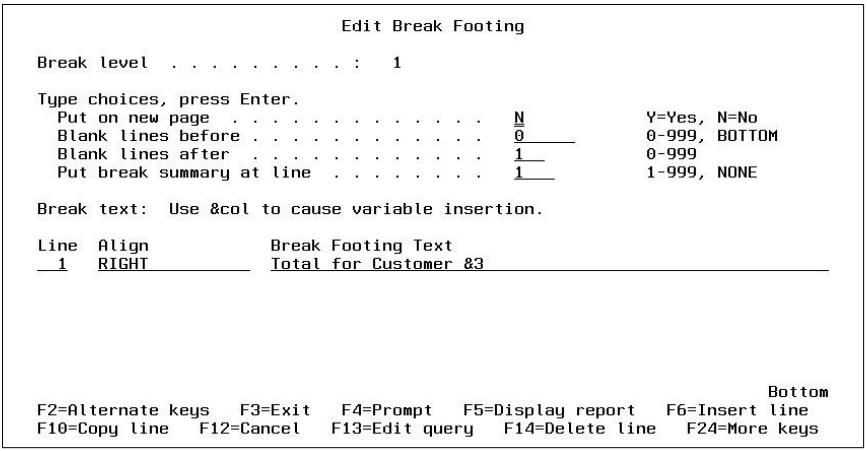


Abbildung 1.16: Break-Level-Text kann auch Spaltendaten enthalten.

Sie werden dazu aufgefordert, die Positionsnummer anzugeben. Setzen Sie diese in einfache Anführungsstriche. Die Ausgabe sollte ähnlich wie in Abbildung 1.17 aussehen.

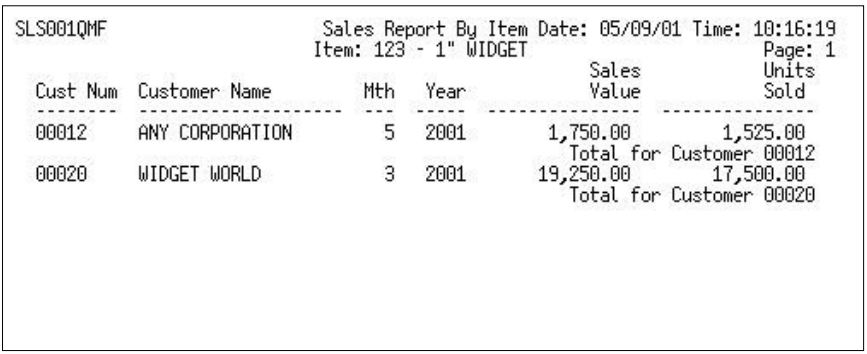


Abbildung 1.17: Verwenden Sie den Bildschirm Work with Query Manager Tables, um QM/400-Tabellen zu definieren.

Query Manager-Tabellen

Der Bildschirm Work with Query Manager Tables (siehe Abbildung 1.18) kann verwendet werden, um Tabellen für die Verwendung mit Query Manager/400 zu definieren.

```
                        Work with Query Manager Tables

Library  . . . . . .  . .   QGPL_____   Name, F4 for list

Type options, press Enter.
  1=Create table   3=Copy table     4=Delete table         5=Display table
  6=Print table    7=Rename table   8=Display definition   9=Add data
  10=Change data   11=Display data

Opt  Table                 Description
_    _________________
__   AGENT
__   AGNTSUM
__   AIPINVFLDS
__   AMES
__   AMY
__   ANDERS
__   ANDERSNS
__   ANDERSONS
__   ANDERSTR
                                                                    More...
F3=Exit   F4=Prompt      F5=Refresh   F11=Display table only   F12=Cancel
F16=Repeat position to   F17=Position to   F24=More keys
```

Abbildung 1.18: Das Ergebnis unseres Beispiel-Query Manager-Berichts.

Dieser Bildschirm kann auch verwendet werden, um Daten in Tabellen einzugeben, die innerhalb oder außerhalb von Query Manager/400 erstellt wurden. Tabelle 1.3 zeigt eine Liste der Optionen, die im Bildschirm Work with Query Manager Tables verfügbar sind und wie diese verwendet werden.

Tabelle 1.3: Optionen für Work with Query Manager Tables	
Option #	**Beschreibung**
1	Verwendung von Create Query Manager Table für die Definition einer neuen Datenbanktabelle
2	Erstellen einer Kopie der ausgewählten Tabelle
3	Löschen einer vorhandenen Tabelle nach Bestätigung in einem Bestätigungsbildschirm
4	Anzeige einer ausgewählten Tabelle unter Verwendung von angegebenen Kriterien
5	Ausdruck einer Liste der ausgewählten Zeilen basierend auf angegebenen Kriterien
6	Umbenennen der ausgewählten Tabelle
7	Anzeige der Tabellendefinition, die nicht aktualisiert werden kann
8	Hinzufügen von Datensätzen zur Tabelle
9	Aktualisierung der Daten in der Tabelle

Wenn in diesem Bildschirm eine Tabelle erstellt wird, wird eine Reihe von Bedienerführungen angezeigt, um Feldnamen, Datentypen (siehe Tabelle 1.4) und Feldlängen für die Spalten in unserer Tabelle zu definieren. Wenn Sie Daten in der Tabelle mit den Optionen 9 und 10 im Bildschirm Query Manager Tables hinzufügen oder aktualisieren, werden die Daten mit Hilfe des in Abbildung 1.19 gezeigten Bildschirms eingegeben.

Tabelle 1.4: Felddatentypen für Query Manager-Tabellen	
Datentyp #	**Beschreibung**
BIGINT	Big Integer
BLOB	Binary large Object
CHARACTER	Beliebiges Zeichen
CLOB	Character Large Object
DATALINK	Datalink
DATE	Datum
DBCLOB	Double-Byte Large Object
DECIMAL	Gepackte Dezimalzahl
Datentyp #	Beschreibung
DOUBLE	Long Floating Poing
FLOAT	Floating Pint
GRAPHIC	Jede Grafik
INTEGER	Long Binary
NUMERIC	Gezont Dezimal
REAL	Short Floating Point
SMALLINT	Short Binary
TIME	Uhrzeit
TIMESTAMP	Zeitstempel
VARCHAR	Zeichen mit variabler Länge
VARGRAPHIC	Variable Graphik

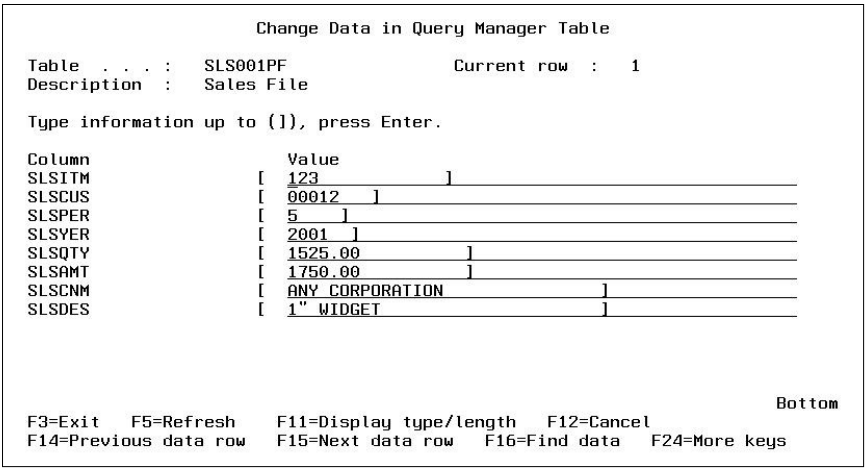


Abbildung 1.19: Mit Hilfe von Work With Query Manager Tables können Dateidaten eingegeben oder aktualisiert werden.

Bevor Sie Datensätze in einer Tabelle aktualisieren, müssen Sie den Datensatz auffinden, der aktualisiert werden soll, indem Sie Find Data im Bildschirm Query Manager Table verwenden. Dieser Bildschirm wird automatisch angezeigt, wenn im Bildschirm Work with Query Manager Tables Option 10 verwendet wird. Mit Hilfe der Operatoren AND und OR können mehrere Bedingungen festgelegt werden. Die Bedingungen können mit Hilfe von rechten und linken Klammern gruppiert werden. Die Vergleichsoperatoren (EQ, NE, GT, GE, LT, LE) werden verwendet, um zu definieren, welcher Typ „Vergleich mit den Spalten“ links im Bildschirm und mit den Werten rechts im Bildschirm erfolgen soll. Nachdem Änderungen an einem ausgewählten Datensatz vorgenommen wurden, drücken Sie die Eingabetaste, um diese Ände-

rungen an der Tabelle zu speichern. F23 kann in diesem Bildschirm gedrückt werden, um den aktuellen Datensatz zu löschen.

Der Zugriff der Benutzer auf die einzelnen Funktionen wird von seiner Zugriffsebene auf die Bibliothek oder die Sammlung, die die Tabelle enthält, kontrolliert.

Query Manager-Profile

Um zu kontrollieren, welche Aktionen die Benutzer in Query Manager/400 ausführen können, wählen Sie im Hauptmenü von Query Manager/400 die Option 10. Der Bildschirm Work with Query Manager Profiles wird angezeigt. Dieser Bildschirm listet alle Profile auf, die aktuell im Query Manager vorhanden sind. Sie können ein neues Profil mit Hilfe von Option 1 in diesem Bildschirm hinzufügen. Wenn Sie Profile hinzufügen oder ändern, haben Sie die Möglichkeit, Informationen einzugeben, wie z.B. die Query Manager-Standardbibliothek des Benutzers, den Standardabfrage-Erstellungsmodus oder auch, ob der Benutzer auf Query Manager Tabellen zugreifen kann. Sie können auch die Anzahl der Datensätze kontrollieren, die zulässig ist, wenn eine Abfrage interaktiv ausgeführt wird, und ob die Anzeige Run Options für diesen Benutzer zulässig ist. Wenn Sie nicht über Systemadministratorberechtigung verfügen, können Sie nur Ihr eigenes Query Manager-Profil ändern.

Um einen Bericht auszuführen, der mit Query Manager/400 erstellt wurde, verwenden Sie den Befehl Start Query Manager Query (STRQMQRY). Dieser Befehl ermöglicht es, sowohl die Abfrage zu definieren, die für die Auswahl unserer Daten verwendet werden wird, als auch das Query Manager-Berichtsformular für die Formatierung der gedruckten Ausgabe. Ein Beispiel für die Verwendung dieses Befehls ist:

```
STRQMQRY QMQRY(libname/qryname) QRYFORM(libname/formname)
         DSPFMT(dispfmt) SETVAR(varname value)
```

Die beiden ersten Befehlsparameter werden verwendet, um die Abfrage und das zu verwendende Berichtsformular auszuwählen. Der Parameter Display Format (DSPFMT) wird angegeben, um auszuwählen, ob die Ausgabe in den Bildschirm gesendet werden sollte (*DISPLAY): zu einem Drucker (*PRINT) oder in eine Ausgabedatei (*OUTFILE). Der Parameter SETVAR legt den Wert der Parameter fest, die in Ihre Abfrage eingefügt werden (&PARMNAME). Der erste Wert für diesen Parameter ist der Parameternamen ohne das vorangestellte kommerzielle Pluszeichen (&). Der zweite Wert für den Parameter SETVAR ist der Wert, der für das ausgewählte Feld verwendet werden soll. Vergessen Sie auf keinen Fall, dass Werte für alphanumerische Felder von einfachen Anführungszeichen (‘ ‘) umschlossen sein müssen.

Query Manager-Prozesse

Am Schluss Ihrer Entdeckungsreise in Query Manager/400 befassen wir uns mit den Prozessen, die Sie in Query Manager einsetzen können. Diese Prozesse erlauben es Ihnen, eine aufeinanderfolgende Anzahl von Query Manager-Befehlen auszuführen – z.B. weil Sie eine Query Management-Abfrage ausführen müssen, um eine Übersichtsdatei zu erstellen und anschließend eine separate Datei für den Ausdruck eines Berichts. Dabei handelt es sich um Befehle, die in einer physischen Quellenteildatei gespeichert und mit dem Befehl Start Query Manager Procedure (STRQMPROC) ausgeführt werden. Diese Befehle verwenden eine Reihe von Query Manager/400-Befehlen, um die entsprechenden Funktionen auszuführen. Bei der Eingabe muss die Groß- und Kleinschreibung beachtet werden und am Anfang und am Ende müssen entweder einfache oder doppelte Anführungszeichen stehen.

Hier die einzelnen Befehle und ihre Funktionen:

- Der Befehl COMMIT führt alle unerledigten Datenbankänderungen aus. Wenn der optionale HOLD-Parameter angegeben wird, bleiben alle offenen SQL-Cursor geöffnet.

- Der Query Manager-Befehl CONNECT wird verwendet, um die Datenbank anzugeben, die für alle folgenden Befehle verwendet werden soll. Dieser Befehl wird im Format CONNECT TO *dbname* angegeben, wobei *dbname* der Namen der Datenbank ist wie in Befehl CRTRDB definiert. Optional kann TO *dbname* vom Parameter-Reset ersetzt werden, von dem die Verbindung mit Remote-Datenbanken beendet wird, und der die lokale Datenbank zur aktuellen Verbindung werden lässt. Zu-

sätzlich kann der Benutzerparameter spezifiziert werden, um einen Benutzernamen für die Datenbank für die Verbindung anzugeben. Der Passwortparameter wird verwendet, um das Passwort für den Benutzer anzugeben, der im Benutzerparameter angegeben wird. Der Benutzerparameter wird entweder als USER – *user_name* – festgelegt, wobei *user_name* mit einer Benutzer-ID für die Datenbank, mit der die Verbindung hergestellt werden soll, ersetzt wird, oder als USER CURRENT für den aktuell angemeldeten Benutzernamen. Der Passwortparameter akzeptiert entweder das mit dem Benutzerparameter verknüpfte Passwort oder optional den Wert NONE.

- In ähnlicher Weise wird der Befehl DISCONNECT verwendet, um die Verbindung mit einer Datenbank zu trennen. Dieser Befehl akzeptiert einen Parameter, der für den Namen der Datenbank angegeben werden muss, deren Verbindung getrennt werden soll, entweder den Sonderwert CURRENT, mit dem die Verbindung mit allen für einen aktuellen Benutzer aktiven Datenbanken getrennt wird, oder den Sonderwert ALL, mit dem die Verbindung mit allen aktuell aktiven Datenbanken beendet wird.

- Der Befehl DISPLAY wird verwendet, um die formatierte Ausgabe der aktiven Daten von der letzten Ausführung des Befehls Run Query oder Run Proc anzuzeigen. Dieser Befehl wird entweder im Format DISPLAY REPORT (FORM=*Library-Formular* angegeben, um die Ausgabe unter Verwendung eines spezifischen Berichtsformulars anzuzeigen, im Format DISPLAY REPORT (FORM=* für die Verwendung des Systemstandards oder im Format DISPLAY SAMPLE für die Anzeige des Berichts mit ausschließlicher Verwendung von Beispieldaten.

- Um ein Query Manager-Objekt von der Datenbank zu entfernen, benutzen Sie den Befehl ERASE. Geben Sie erase gefolgt vom Typ des zu entfernenden Objekts (TABLE, FORM, PROC, QUERY) sowie den qualifizierten Namen des zu entfernenden Objekts an. Der optionale Parameter CONFIRM YES kann ebenfalls angegeben werden, um dafür zu sorgen, dass Query Manager den Benutzer dazu auffordert, den Befehl zu bestätigen.

- Der Befehl EXPORT wird verwendet, um die Quelle für ein Query Manager Objekt vom Typ FORM, PROC oder QUERY unter Verwendung des Formats EXPORT FORM *Formularname* TO *Library.Dateiname(Teildatei)* abzurufen. Optional kann der Sonderwert *FIRST verwendet werden, um die Ausgabe in die erste Teildatei der gewählten physischen Source-Datei einzufügen.

- Der Befehl IMPORT verwendet die Teildatei, die mit dem Befehl EXPORT erstellt wurde, um ein neues Query Manager-Objekt zu erstellen. Der Befehl IMPORT verwendet die gleichen Parameter wie im Befehl EXPORT.

- Der Befehl PRINT wird verwendet, um die Ausgabe des Berichtsformulars, der Abfrage oder des Prozesses zu einem angegebenen Drucker oder zum Standarddrucker für den aktuellen Job zu senden. Der Befehl PRINT FORM FORMLIB.FORM1 sendet die Ausgabe des zuletzt eingegebenen Befehls RUN QUERY oder RUN PROC zum Standarddrucker unter Verwendung des Formulars mit dem Namen FORM1 in der Bibliothek mit dem Namen FORMLIB. Sie können diesen Befehl auch im Format PRINT REPORT verwenden, das die WIDTH-Parameter verwendet, um die Zeilenbreite in Zeichen anzugeben; LENGTH, um die Seitenlänge in Zeilen zu definieren; FORM, um das zu verwendende

Formular zu definieren; DATATIME, um anzugeben, ob die gedruckte Ausgabe mit Datum und Zeitpunkt ergänzt werden soll; und PAGENO, um auszuwählen, ob die Seitennummer dem Bericht hinzugefügt werden soll. PRINT REPORT kann verwendet werden, um die Ausgabe zu steuern, indem unterschiedliche Seitengrößen und optionale Informationen zugelassen werden. Für das oben genannte Beispiel verwenden Sie folgenden Befehl, um den zuvor verwendeten Befehl PRINT FORM zu ersetzen:

```
PRINT REPORT (PRINTER=PRT01 WIDTH=80 LENGTH=66 DATETIME=NO
      PAGENO=NO FORM=FORMLIB.FORM1
```

- Der Befehl RELEASE ermöglicht es Ihnen, eine Datenbankverbindung freizugeben, bevor der Befehl COMMIT eingegeben wird, mit dem die Datenbankverbindung getrennt wird. Um diesen Befehl zu definieren, geben Sie RELEASE *dbname* ein oder Sie geben einen Sonderwert CURRENT oder ALL ein. Beide dieser Werte funktionieren wie im Befehl DISCONNECT.

- Der Befehl RUN wird verwendet, um entweder eine Query Manager Query (RUN QUERY) oder eine Query Manager Procedure (RUN PROC) auszuführen. Dem Befehl folgt der Namen des auszuführenden Objekts (RUN PROC QRYPROC). Optional kann für den Befehl RUN QUERY der Parameter FORM= *formname* angegeben werden, um das Formular zu definieren, das mit dieser Abfrage verwendet werden soll, und DISPLAY=YES|NO, um dem Query Manager mitzuteilen, ob er die Ausgabe im Bildschirm anzeigen soll, wenn diese Prozedur interaktiv ausgeführt wird.

- Der Befehl SAVE DATA wird verwendet, um eine Kopie der aktiven Daten vom zuletzt eingegebenen Befehl RUN QUERY in eine

Datendatei zu speichern. Dieser Befehl verwendet das Format SAVE DATA AS FILENAME (CONFIRM=YES ACTION=REPLACE. Dieser Befehl fügt die Ergebnisse des zuletzt eingegebenen Befehls RUN QUERY in eine Datei mit der Bezeichnung FILENAME ein, fordert den Benutzer beim Aufruf zur Eingabe auf und ersetzt die Datei, wenn diese Datei bereits vorhanden ist.

- SET CONNECTION ermöglicht es Ihnen zu definieren, welche der aktuell verbundenen Datenbanken zur aktuellen Verbindung werden soll.

- SET GLOBAL wird verwendet, um den Wert einer Query Manager-Variable von einer Query Manager-Prozedur aus zu definieren. Ein Beispiel für die Verwendung ist SET GLOBAL CompNm='ABC Corporation'.

Indem Sie Kombinationen dieser Befehle von einer Query Manager-Prozedur aus ausführen, erhalten Sie die Möglichkeit, voll funktionsfähige komplexe Berichtsprogramme innerhalb von Query Manager/400 zu erstellen.

Wie Sie sehen, erhalten Sie durch Nutzung der halben Leistungsfähigkeit eines RPG-Programms und der doppelten Funktionalität von Query/400 mit Query Manager/400 alle Tools, die Sie benötigen, um die Arbeitszeit bei der Erstellung von neuen Berichten auf der iSeries400 zu reduzieren.

2

Entwicklung von „Green Screen“ GUI-Anwendungen

Anwendungen mit einer so genannten Graphical User Interface (GUI) sind so populär, dass man den Eindruck gewinnen könnte, dass Green-Screen-Anwendungen etwas unzeitgemäß sind. Sie können die Attraktivität dieser Programme jedoch erhöhen, indem Sie Anwendungen erstellen, die erweiterte Display-DDS-Funktionen nutzen. Eine Sammlung von Display-File-Schlüsselwörtern ermöglicht es Ihnen, in einer Green-Screen-Anwendung Fenster, Dropdown-Menüs sowie auch Schaltflächen und Optionsschaltflächen zu erstellen. Der Vorteil davon ist, dass die meisten Funktionen sogar den Einsatz einer Maus unterstützen – zum Beispiel kann der Anwender, wenn die Display File einer Anwendung ein Schaltflächen-Objekt enthält, dieses Objekt aktivieren, indem er einfach mit der Maus darauf klickt. In diesem Kapitel befassen wir uns mit den einzelnen Objekten und ihrer Verwendung.

Display-File-Fenster

Fenster und Pulldown-Menüs sind das „A und O“ der GUI-Anwendungen in den meisten modernen PC-Betriebssystemen. In OS/400 ermöglicht Ihnen die Windows-Technik, einen angegebenen Bereich des Bildschirms für erforderliche Daten auszuwählen und in einem begrenzten Bereich anzuzeigen. Dies ist aus ästhetischen ebenso wie aus funktionalen Gründen sinnvoll. Wenn Sie ein Fenster für die Suche oder Anzeige von zusätzlichen Informationen aufrufen können, während die Informationen in Ihrem Hauptfenster noch sichtbar sind, wird die Dateneingabe einfacher.

Von einem ästhetischen Standpunkt aus gesehen erhalten Ihre Anwendungen durch die Verwendung von erweiterten Anzeigefenstern eine modernere Schnittstelle. Popup-Menüs lassen außerdem Ihre Anwendung nicht nur besser aussehen, sondern sie bieten eine neue Methode für die Kontrolle des Programmflusses. Dies geschieht, indem Sie Funktionen verlagern, die normalerweise vom Benutzer gesteuert werden, indem er die Funktionstaste für ein Menü drückt. Wenn Sie jemals eine Anwendung gesehen haben, die vollständig ohne Funktionstasten konzipiert ist, wissen Sie, wie nützlich das sein kann.

Die Windows-Technik ermöglicht es Ihnen, allgemeine Funktionen in Menüs zu organisieren, um die Steuerung von Anwendungsfunktionen logischer zu gestalten. Die Kombination dieser beiden Ziele kann Ihnen wirklich dabei helfen, Ihrer Anwendung das Aussehen und Verhalten einer GUI zu verleihen. Als Erstes erfahren Sie, welche Schritte erforderlich sind, um ein Display-Fenster zu erstellen. Diese Funktionalität steht eigentlich in OS/400 schon seit Jahren zur Verfügung, aber ich staune immer wieder, wie Display-

File-Fenster in der Praxis vernachlässigt werden. Diese Fenster werden grundsätzlich auf die gleiche Weise wie Standard-Display-Datensätze verwendet, davon ausgenommen ihr Verhalten bei der Anzeige innerhalb einer Client Access Display-Sitzung oder bei der Anzeige auf einem anderen Terminal-Client, der erweiterte Anzeigefunktionen unterstützt, denn hier werden sie fast dreidimensional dargestellt.

Maßgeblich bei der Definition eines Fensters ist das Record-Level DDS-Schlüsselwort WINDOW. Dieses Schlüsselwort wird im folgenden Format angegeben:

```
WINDOW(y-Position, x-Position, Höhe, Breite)
```

Die angegebenen vier Parameter können innerhalb der Display File statisch definiert werden, oder Sie können Felder vom Typ *Program-to-System* verwenden, mit denen diese Werte von Ihrem Anwendungsprogramm aus dynamisch geändert werden können, anstatt statische Werte zu verwenden. Ein Program-to-System-Feld wird verwendet, um Werte von Ihrem Programm an einen Parameter für ein DDS-Schlüsselwert zu übergeben. In diesem Fall können die horizontale und die vertikale Position, Höhen- und Breitenparameter mit Hilfe von Program-to-System-Feldern definiert werden.

Sie können ohne großen Aufwand einen Titel in den Anfang oder in das Ende des Fensters einfügen, indem Sie WDWTITLE angeben. Dieses Schlüsselwort wird wie folgt spezifiziert:

```
WDWTITLE([Text] [Textfarbe] [Anzeigeattribute] [*CENTER |
*LEFT | *RIGHT] [*TOP | BOTTOM])
```

Dieses Schlüsselwort auf Datensatzebene (Record-Level) kann mehrfach angegeben werden. Das ermöglicht es Ihnen, Titeltext für den Anfang und für das Ende des Fensters oder – bei sehr großen Fenstern – in jeder Ecke des Fensters zu definieren. Der erste Parameter wird verwendet, um den eigentlichen Text des Titels festzulegen. Dieser Text wird entweder in einfache Anführungszeichen gesetzt oder das Program-to-System-Feld wird verwendet.

Die Farbe des Titeltexts wird mit dem Parameter text COLOR definiert. Zuerst wird *COLOR angegeben, gefolgt von dem Code für die Farbe, der aus drei Buchstaben besteht. Diese Codes entsprechen den Codes, die für Display-File-Schlüsselwort COLOR angegeben werden und die für die Festlegung der Farbe eines Feldes benutzt werden. Tabelle 2.1 enthält eine Liste der Farben und den Farbcode für die Definition der Titelfarbe.

Tabelle 2.1: Farbcodes für Display File Schlüsselwörter

Code	Farbe	Code	Farbe
BLU	Blau	CYN	Cyan
RED	Rot	MGN	Magenta
YLW	Gelb	WHT	Weiß
GRN	Grün	PUR	Purpur

Das Anzeigeattribut wird durch die Angabe von (*DSPATR definiert, gefolgt vom Attributcode für die Anzeige, der aus zwei Buchstaben besteht – zum Beispiel HI für „High Intensity“. Diese Codes entsprechen wiederum den Codes im Display-File-Schlüsselwort

DSPATR. Mit den Werten *CENTER, *LEFT, und *RIGHT wird die horizontale Positionierung des Titels definiert. Die Werte für *TOP und *BOTTOM werden verwendet, um die vertikale Positionierung des Texts zu definieren. Wenn keiner dieser Werte angegeben wird, wird der Text standardmäßig ganz oben in der Mitte des Fensters angezeigt.

Das Aussehen des Fensterrands deklarieren Sie mit dem Schlüsselwort WDWBORDER. Dieses Record-Level-Schlüsselwort wird wie folgt definiert:

```
WDWBORDER([color] [display attribute] [border characters])
```

Die Farb- und Anzeigeattribut-Parameter werden auf die gleiche Weise wie beim Schlüsselwort WDWTITLE definiert. Der Parameter für das Randzeichen bestimmt dabei, welche Zeichen für die jeweiligen Teile des Fensterrands verwendet werden, wenn das Fenster auf einem Gerät angezeigt wird, das keine erweiterten Anzeigefunktionen unterstützt. Dieser Parameter wird definiert als (*CHAR, gefolgt von einer Zeichenkette, die die zu verwendenden Zeichen enthält – in dieser Reihenfolge: für oben links, oben Mitte, oben rechts, linke Seite, rechte Seite, ganz unten links, ganz unten in der Mitte und unten rechts. Ein Beispiel für dieses Format ist WDWBORDER((*CHAR '/-\||\-/'), bei dem Zeilenzeichen für den oberen und seitlichen Bereich sowie ein Querstrich (/ – nach links gerichteter Querstrich für die Ecken) verwendet werden. Diese drei Schlüsselwörter (WINDOW, WDWTITLE und WDWBORDER) helfen dabei, einfache Fenster in Ihrer Display File zu definieren.

Um alle Teile dieses Puzzles zusammenzusetzen, erstellen wir zur Übung eine einfache Anwendung mit einem Fenster. Die DDS-Source für die Display-File wird in Abbildung 2.1 gezeigt. Diese Display-File zeigt Datum und Uhrzeit in einem festgelegten Bereich im Bildschirm an.

```
     *************************************************************************
     *
     * Befehle erstellen: CRTDSPF FILE(library/WIN001RG)
     *                            SRCFILE(library/QDDSSRC)
     *
     *************************************************************************
A            R DATETIME
A                                        WINDOW(&X &Y 6 18)
A                                        WDWTITLE((*TEXT 'DATE & TIME') +
A                                        *CENTER *TOP)
A                                     2  1'DATE:'
A                                     2  7DATE(*SYS) EDTCDE(Y)
A                                     4  1'TIME:'
A                                     4  7TIME
A              Y               2S 0P
A              X               3S 0P
```

Abbildung 2.1: DDS-Source für die Erstellung eines einfachen Display File-Fensters.

Das Fenster wird mit Hilfe der Program-to-System-Variablen *X* und *Y* positioniert. Damit können wir die Position des Fensters von innerhalb der Anwendung aus festlegen.

Abbildung 2.2 zeigt den ILE-RPG-Code für die Anzeige dieses Fensters. Unser Fenster erhält mit Hilfe des DDS-Schlüsselworts WDWTITLE einen Titel.

```
 ****************************************************************************
 *
 * Befehl erstellen: CRTBNDRPG PGM(library/WIN001RG)
 *                             SRCFILE(library/QRPGLESRC)
 *
 ****************************************************************************
FWIN001DS  CF   E               WORKSTN
C     *ENTRY          PList
C                     Parm                    XPOS            15 5
C                     Parm                    YPOS            15 5
C*
C                     If        XPOS=0 or YPOS=0
C                     Eval      X=20
C                     Eval      Y=10
C                     Else
C                     Z-Add     XPos          X
C                     Z-Add     YPos          Y
C                     EndIf
C*
C                     Exfmt     DateTime
C*
C                     Eval      *INLR=*On
```

Abbildung 2.2: Diese RPG-Source wird für die Anzeige unseres Beispielfensters verwendet.

Dieses Programm akzeptiert Parameter, die verwendet werden, um den Fensterdatensatz in der Anzeige zu positionieren. Wenn Nullwerte für einen dieser Parameter festgelegt werden, wird eine Standardposition verwendet. Als Resultat wird das aktuelle Datum und die aktuelle Uhrzeit angezeigt.

Abbildung 2.3 zeigt, wie dieses Fenster in Ihrer Anzeige erscheint.

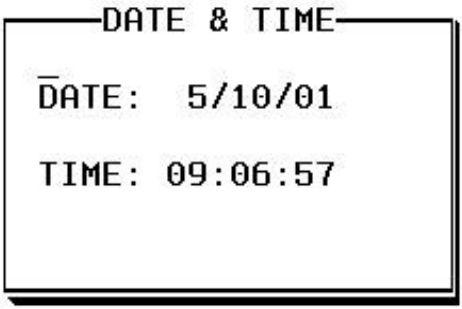


Abbildung 2.3: Beispielfenster für Datum und Uhrzeit

Hierbei handelt es sich um ein einfaches Beispiel für die Funktionen, die mit einem Display-File-Fenster implementiert werden können. Zu den wichtigsten Einsatzmöglichkeiten gehört z.B. die Kombination mit einer Subfile. Damit erhalten Sie ein Objekt, das große Ähnlichkeit mit einem Listenfeld aufweist.

Das wird erreicht, indem ein Fenster erstellt wird, das als Subfile Control Record fungiert. Der verknüpfte Subfile-Datensatz wird innerhalb des Fensters angezeigt.

Ein Beispiel für eine Subfile-Display-File mit Fenster sehen Sie in Abbildung 2.4.

```
  *=====================================================================
  *   Kompilierung:
  *
  *     CRTDSPF  FILE(XXX/LST001DS) SRCFILE(XXX/QDDSSRC)
  *
  *=====================================================================
A                                     DSPSIZ(24 80 *DS3)
A                                     CA03
A        R LISTSF                     SFL
A                                     SFLNXTCHG
A          SEL           1A  B  1  1VALUES('1')
A          NCKNAM       22A  H
A          LSTITM       30A  O  1  3
A        R LIST                       SFLCTL(LISTSF)
A  50                                 SFLDSP
A  50                                 SFLDSPCTL
A N50                                 SFLCLR
A N50                                 SFLDLT
A  50                                 SFLEND(*SCRBAR *SCRBAR)
A                                     SFLSIZ(0100)
A                                     SFLPAG(0015)
A                                     WINDOW(3 22 17 35)
A                                     WDWTITLE((*TEXT 'Select State'))
A        R LISTNICK
A                                     WINDOW(8 15 6 37)
A                                     WDWTITLE((*TEXT 'State Nickname'))
A                                  2  1'State:'
A          STTABB        2A  O   2  8
A          STNAME       27A  O   2 11
A                                  4  1'Nickname:'
A          NCKNAM       22A  O   4 14
```

Abbildung 2.4: Die DDS für eine Subfile, die innerhalb eines Fensterdatensatzes angezeigt wird.

Die verknüpfte ILE RPG Source für dieses Beispiel sehen Sie in Abbildung 2.5. Dieses einfache Programm zeigt eine Liste der U.S.-Staaten an und erlaubt es dem Benutzer, einen Staat auszuwählen, indem er eine 1 neben den gewünschten Namen angibt. Sicherlich wird Ihnen auffallen, dass wir für WINDOW das Schlüsselwort SFLEND(*SCRBAR *SCRBAR) angeben. Dieses Schlüsselwort ersetzt + or MORE, das normalerweise verwendet wird, um in Form einer grafischen maussteuerbaren Scroll-Leiste anzuzeigen, dass diese zusätzlichen Subfile-Seiten vorhanden sind.

Eine maussteuerbare Scroll-Leiste ermöglicht es Ihnen, die Subfile neu zu positionieren, ohne die Nach-oben-/Nach-unten-Tasten zu verwenden. Statt dessen können Sie mit der Maus auf die Nach-oben-/Nach-unten-Symbole klicken, oder Sie ziehen einfach die Scroll-Leiste selbst nach unten.

```
 *=============================================================
 *   Kompilierung:
 *         CRTBNDRPG PGM(XXX/LST001RG) SRCFILE(XXX/QRPGLESRC)
 *
 *=============================================================
FLST001DS  CF   E             WORKSTN SFILE(LISTSF:RRN)
D Stte            S             52    DIM(50) CTDATA PERRCD(1)
D RRN             S              5  0
D EOFLST          S                   LIKE(*INLR)

 * Datenstruktur für die Trennung von Staatenkürzel und Namen
D DataStr         DS            52
D  LSTITM                  1     30
D  SttAbb                  1      2
 * Freier Bereich für die Anzeige
D  StName                  4     30
D  NckNam                 31     52
 *
 * Subfile löschen
```

Abbildung 2.5: Dieses Programm verwendet eine Windows-Subfile, um eine Liste der US-Staaten zu erstellen (Teil 1 von 3).

```
 *
C                   DoW       *INLR=*Off
C                   Eval      *IN50=*Off
C                   Eval      RRN=0
C                   Clear                   SEL
C                   Write     LIST
 *
 * Subfile laden
 *
C                   Do        50            RRN
C                   Eval      DataStr=Stte(RRN)
C                   Write     LISTSF
C                   EndDo
C                   Eval      *IN50=*On
 *
 * Liste anzeigen
 *
C                   Exfmt     LIST
C                   If        *INKC=*On
C                   Eval      *INLR=*On
C                   Else
C                   Eval      EOFLST=*Off
C                   DoW       EOFLST=*Off
C                   ReadC     LISTSF
C                   Eval      EOFLST=%EOF(LST001DS)
C                   If        EOFLST=*Off And SEL='1'
C                   Exfmt     LISTNICK
C                   Leave
C                   EndIF
C                   EndDo
C                   EndIF
C                   EndDo
 ** Tabelle mit Staatennamen
AL Alabama               Yellowhammer State
AK Alaska                The Last Frontier
AZ Arizona               The Grand Canyon State
AR Arkansas              The Natural State
CA California            Golden State
CO Colorado              Centennial State
```

Abbildung 2.5: Dieses Programm verwendet eine Windows-Subfile, um eine Liste der US-Staaten zu erstellen (Teil 2 von 3).

```
CT Connecticut          Constitution State
DE Delaware             First State
FL Florida              Sunshine State
GA Georgia              Peach State
HI Hawaii               Aloha State
ID Idaho                Gem State
IL Illinois             Prairie State
IN Indiana              Hoosier State
IA Iowa                 Hawkeye State
KS Kansas               Sunflower State
KY Kentucky             Bluegrass State
LA Louisiana            Pelican State
ME Maine                Pine Tree State
MD Maryland             Old Line State
MA Massachusetts        Bay State
MI Michigan             Wolverine State
MN Minnesota            North Star State
MS Mississippi          Magnolia State
MO Missouri             Show Me State
MT Montana              Treasure State
NE Nebraska             Cornhusker State
NV Nevada               The Silver State
NH New Hampshire        Granite State
NJ New Jersey           Garden State
NM New Mexico           Land of Enchantment
NY New York             Empire State
NC North Carolina       Tar Heel State
ND North Dakota         Roughrider State
OH Ohio                 Buckeye State
OK Oklahoma             Sooner State
OR Oregon               Beaver State
PA Pennsylvania         Keystone State
RI Rhode Island         The Ocean State
SC South Carolina       Palmetto State
SD South Dakota         Mount Rushmore State
TN Tennessee            Volunteer State
TX Texas                Lone Star State
UT Utah                 The Beehive State
VT Vermont              Green Mountain State
```

Abbildung 2.5: Dieses Programm verwendet eine Windows-Subfile, um eine Liste der US-Staaten zu erstellen (Teil 3 von 3).

Eine interessante Eigenschaft der Scroll-Leiste ist ihr Verhalten, wenn Sie auf die Pfeile „Nach oben“ und „Nach unten“ klicken. In diesem Fall wird die Subfile jeweils um einen Datensatz neu positioniert anstatt um eine volle Seite. Ein Beispiel für dieses Fenster sehen Sie in Abbildung 2.6.

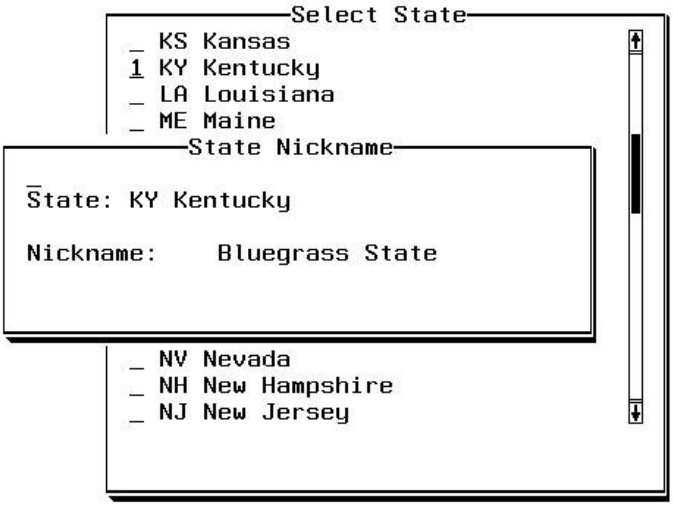


Abbildung 2.6: Eine Fenster-Subfile kann für die Erstellung eines fensterähnlichen Listenfelds verwendet werden.

Programmierung von Menüs

Zusätzlich zu den Fenstern ermöglichen es Ihnen Popup-Menüs, Ihren Anwendungen ein benutzerfreundliches „Look and Feel“ zu verleihen. Popup-Menüs bestehen aus zwei Komponenten. Die erste Komponente ist die Menüleiste mit einer Zeile, die mehrere Optionen enthält. Jede dieser Optionen kann mit einem Pulldown-Datensatz verknüpft werden, der den „Dropdown-Effekt“ erzielt, der in den meisten Anwendungen in einer grafischen Anwendung zum Einsatz kommt.

Menüleisten basieren auf dem Record-Level DDS-Schlüsselwort (MNUBAR). Dieses Schlüsselwort informiert darüber, dass das Datensatzformat eine Menüleiste ist und ermöglicht Ihnen die Festlegung, ob eine Trennzeile unterhalb der Menüleiste angezeigt werden soll. Das Schlüsselwort wird wie folgt angegeben:

```
MNUBAR(*SEPARATOR or *NOSEPARATOR)
```

Beide Parameter sind optional, wenn sie fehlen, ist *SEPARATOR die Standardoption.

Jede Menüleiste muss ein Feld enthalten, für das mindestens ein Schlüsselwort Menubar Choice (MNUBARCHC) definiert sein muss. Dieses Schlüsselwort wird wie folgt definiert:

```
MNUBARCHC(Option #, Pull Down Record, Option Text, Return Value Field)
```

Der erste Parameter wird verwendet, um zu definieren, welcher Wert an das Feld zurückgegeben wird, das das Schlüsselwort MNUBARCHC enthält, wenn dieses Menü ausgewählt wird. Der

nächste Parameter wird verwendet, um das Pulldown-Menü-Datensatzformat für dieses Menü mit der Menüleistenoption zu verknüpfen. Der Parameter „Option Text“ erlaubt es Ihnen, anzugeben, was im Bildschirm für diese Menüoption angezeigt wird. Sie können *Tastenkombinationen* für Menüleistenoptionen definieren, indem Sie das Größer-als-Zeichen (>) für die Taste einfügen, die als Shortcut verwendet werden soll. Wenn Sie z.B. > als Optionstext verwenden, wird dieser im Bildschirm als Datei angezeigt. Wenn der Menüleistendatensatz aktiv ist, kann diese Option ausgewählt werden, indem einfach die F-Taste gedrückt wird.

Der letzte Parameter – Return Value Field – ist optional und wird verwendet, um ein verborgenes numerisches Feld festzulegen, dessen Datentyp Y lautet, mit einer Länge von 2 und mit 0 Dezimalstellen. Wenn dieser Parameter festgelegt wird, wird die Steuerung an das Anwendungsprogramm zurückgegeben, falls diese Option aktiviert wird. Dieses Feld wird dann die Menüleisten-Optionsnummer enthalten, die ausgewählt wurde. Wenn dieser Parameter nicht angegeben wird, wird automatisch das Pulldown-Menü angezeigt, das mit der Menüleistenoption verknüpft ist. Wenn die Steuerung an das Programm zurückgegeben wird, enthält das Return-Value-Feld 0. Abbildung 2.7 enthält ein Beispiel für das Menüleisten-Datensatzformat.

```
A      R BAR1                         MNUBAR
A        MNUFLD         2Y 0B  1  2
A                                     MNUBARCHC(1 FILE '>File ')
A                                     MNUBARCHC(2 EDIT '>Edit ')
A                                     MNUBARCHC(3 VIEW '>View ')
A                                     MNUBARCHC(4 TOOLS '>Tools ')
A                                     MNUBARCHC(5 HELP '>Help ' &RTNVAL)
A        RTNVAL         2Y 0H
```

Abbildung 2.7: Dies ist ein einfaches Menüleisten-Datensatzformat.

In diesem Beispiel wird eine Gruppe von Menüleistenoptionen konfiguriert, die den Optionen in Microsoft Word ähneln. Dieses Menü enthält fünf Optionen: File, Edit, View, Tools und Help. Jede dieser Optionen ist mit einem Pulldown-Datensatzformat verknüpft.

Die Option „Help“ verwendet ein Return-Value-Feld für die Rückgabe der Steuerung an die Anwendung, wenn das Menü „Help“ aktiviert wird. Wir haben außerdem für jede der Optionen Tastenkombinationen definiert. Die Pulldown-Menüs selbst werden in einem separaten Datensatz mit Hilfe des Schlüsselworts PULLDOWN definiert. Dieses Schlüsselwort wird auf Datensatzebene verwendet und kennzeichnet einen Datensatz als Pulldown-Menü. Wenn für ein Datensatzformat eine WRITE-Operation ausgeführt wird, wird der Datensatz erst dann angezeigt, wenn das erforderlich ist, basierend auf der Option, die in der Menüleiste ausgewählt wurde.

Der Pulldown-Datensatz selbst ist grundsätzlich ein Fensterdatensatz, dessen Größe und Position dynamisch auf Grundlage der Anzahl von Optionen und der Länge der Optionen bestimmt wird, die im Pulldown angezeigt werden sollen. Der Datensatz, der das Schlüsselwort PULLDOWN enthält, kann kein Schlüsselwort WINDOW enthalten. Das Pulldown-Menü kann Optionen enthalten, die mit Hilfe der DDS-Schlüsselwörter SFLSNGCHC, SFLMLTCHC, SNGCHCFLD und MLTCHCFLD definiert werden. SFLSNGCHC und SFLMLTCHC werden für einen Subfile-Datensatz verwendet, für den der Pulldown-Datensatz als Subfile-Steuerdatensatz agiert. Dies ermöglicht es Ihnen, die Menüoptionen dynamisch von innerhalb Ihrer Anwendung aus zu erstellen.

Mit dem Schlüsselwort Subfile Single Choice (SFLSNGCHC) wird festgelegt, dass nur eine Option innerhalb der Subfile ausgewählt werden kann – bzw. in diesem Fall das Pulldown-Menü. Dieses Schlüsselwort wird im folgenden Format definiert:

```
SFLSNGCHC (*RSTCSR/*NORSTCSR. *SLTIND/*NOSLTIND,
           *AUTOSLT/*NOAUTOSLT/*AUTOSLTENH)
```

Der erste Parameter informiert darüber, ob der Cursor auf ein Feld beschränkt werden sollte. Wenn *RSTCSR angegeben wird, kann der Cursor nicht mit Hilfe der Pfeiltasten außerhalb des Felds verschoben werden, während *NORSTCSR es Ihnen erlaubt, den Cursor außerhalb des Felds mit Hilfe der Pfeiltasten zu verschieben. Der nächste Parameter bestimmt, ob die aktuell ausgewählte Option angegeben wird. Wenn *SLTIND angegeben wird, wird ein Optionsschalter verwendet, um festzulegen, welche Option aktuell ausgewählt ist. Wenn *NOSLTIND verwendet wird, wird die ausgewählte Option einfach markiert.

Der dritte Parameter wird verwendet, um anzugeben, wie die Optionen ausgewählt werden. Wenn *AUTOSLT angegeben wird, wird durch Drücken der Eingabetaste die Option zurückgegeben, die aktuell den Auswahl-Cursor für die Rückgabe enthält. Wenn *NOAUTOSLT ausgewählt ist, müssen Sie als Erstes die Option wählen, indem Sie entweder die Leertaste drücken oder doppelt mit der Maus klicken, um diese Option zu aktivieren. *AUTOSLTENH legt fest, dass die Option nur dann *AUTOSLT verwendet, wenn der Bildschirm auf einer Workstation angezeigt wird, die erweiterte Anzeigefunktionen unterstützt. Das Schlüsselwort Subfile Multiple Choice (SFLMLTCHC) funktioniert weitgehend wie das Schlüsselwort SFLSNGCHC – allerdings mit einigen Einschränkungen.

Erstens können für eine Subfile, die mit dem Schlüsselwort SFLMLTCHC definiert wird, mehrere Optionen ausgewählt werden. Aus diesem Grund gelten die Autoselect-Optionen nicht. Dieses Schlüsselwort wird im folgenden Format kodiert:

```
SFLMLTCHC(Feld # der ausgewählten Optionen, *SLTIND/*NOSLTIND,
                RSTCSR/NORSTCSR)
```

Der erste Parameter wird verwendet, um an das Programm die Anzahl der ausgewählten Optionen zurückzugeben. Dieser Parameter ist der Namen des verborgenen numerischen Felds mit der Länge von 4 Stellen – ohne Dezimalstellen. Die beiden anderen Parameter werden wie beim Schlüsselwort SFLSNGCHC definiert. Wenn die Option *SLTIND verwendet wird, wird anstatt der Optionsschaltfläche, die für SFLSNGCHC angezeigt wird, ein Kontrollkästchen angezeigt. Beide Schlüsselwörter – SLFSNGCHC und SFLMLTCHC – werden auf Datensatzebene in einem Subfile-Steuerdatensatz für einen Subfile-Datensatz mit nur einem Feld für die Anzeige festgelegt. Dieses Feld wird als die Auswahloption für das Pulldown-Menü verwendet.

Bei Verwendung des Schlüsselworts SFLMLTCHC können Sie gleichzeitig mehrere Datensätze auswählen. Der Subfile-Datensatz für die einzelnen ausgewählten Datensätze muss ein einzelnes Feld für die Anzeige sowie ein verborgenes einstelliges Feld ohne Dezimalstellen enthalten, das als Optionssteuerfeld verwendet wird. Dieses Feld wird als erstes Feld im Subfile-Datensatz definiert und muss das Schlüsselwort SFLCHCCTL enthalten. Das Format dieses Schlüsselworts lautet:

```
SFLCHCCTL(msg-id msg_file_lib/msg_file)
```

Die ausgewählte Option (oder Optionen) wird mit Hilfe des Befehls Read Changed Record (READC) RPG angegeben, gemeinsam mit dem Wert des Felds, das im Schlüsselwort SFLCHCCTL definiert ist. Dieses Feld wird eine 0 enthalten, wenn die Option nicht ausgewählt ist, und eine 1, wenn das der Fall ist. Optionen können auch mit Hilfe dieses Felds bei der Ausgabe gesteuert werden. Dies geschieht, indem der Wert des Felds, das das Schlüsselwort SFLCHCCTL enthält, vor dem Schreiben des Datensatzes festgelegt wird, indem einer der Werte in Tabelle 2.2 verwendet wird.

Tabelle 2.2: Steuerwerte für Schlüsselwort SFLCHCCTL

Wert	Funktion
0	Option ist verfügbar, aber aktuell nicht ausgewählt.
1	Option ist verfügbar und ausgewählt.
2	Option ist nicht verfügbar. Der Mauszeiger kann zu der Option verschoben werden, wenn für diese Hilfe definiert wurde.
3	Option ist nicht verfügbar. Mauszeiger kann zur Option verschoben werden.
4	Option ist nicht verfügbar. Mauszeiger kann nicht zur Option verschoben werden.

Die Parameter für das Schlüsselwort SFLCHCCTL werden angegeben, um die Nachrichtennummer und optional die Bibliothek und die Nachrichtendatei zu definieren, die die anzuzeigende Nachricht enthält, wenn eine nicht verfügbare Option ausgewählt wur-

de. Die DDS-Source für die Display-File MNU001DS und die ILE RPG-Quelle für das Programm MNU001RG können von der in der Einleitung angegebenen Webseite heruntergeladen werden.

In diesem Beispiel wird eine einfache Menüleiste mit drei Optionen erstellt: File, Edit und View. Jede dieser Optionen verfügt über ein Pulldown-Menü, von dem das Schlüsselwort SFLCHCCTL verwendet wird, um die Menüoptionen zu erstellen. Die Subfiles für diese Pulldown-Menüs werden beim Start des Programms geladen und automatisch mit der verknüpften Menüleiste angezeigt, ohne dass ein Programmeingriff erforderlich ist. Der Wert des Felds MNUFLD, für das in der Display File das Schlüsselwort MNUBARCHC enthalten ist, wird anschließend verwendet, um zu bestimmen, welches Menü ausgewählt wurde.

Als Nächstes wird Read Changed Record (READC) für den Pulldown-Subfile-Datensatz ausgeführt, der mit der ausgewählten Menüleistenoption verknüpft ist. Die erforderliche Subroutine für die ausgewählte Option wird auf Grundlage des Werts des Felds OPTION gestartet. Dieses Feld enthält den Text, der im Pulldown-Menü selbst angezeigt wird. Wir geben daher das Zeichen > an, das verwendet wird, um die Tastenkombination zu definieren, wenn die Prüfung der gewählten Option erfolgt. Erstellen Sie die Display-File mit dem Befehl Create Display File (CRTDSPF) und kompilieren Sie das ILE RPG-Programm mit dem Befehl Create Bound RPG Program (CRTBNDRPG).

Als Nächstes führen Sie das Programm aus, indem Sie CALL PGM(MNU001RG) ausführen. Abbildung 2.8 zeigt ein Beispiel für den angezeigten Bildschirm.

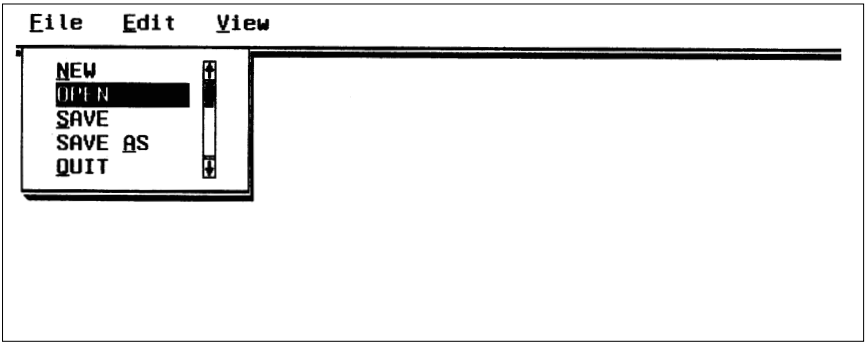


Abbildung 2.8: Dieses Programm MNU001RG zeigt ein Beispiel für Pulldown-Menüs, die mit Hilfe von Subfiles erstellt wurden.

Der Vorteil der Verwendung von Subfiles bei der Erstellung eines Pulldown-Menüs ist, dass Sie Ihr Menü aus einer ILE RPG-Anwendung heraus dynamisch mit Optionen ergänzen können. Es ist auch möglich, die Optionen für das Pulldown-Menü ohne Subfile-Datensätze zu definieren. Das Schlüsselwort Single Choice Field (SNGCHCFLD) wird verwendet, um ein einzelnes Optionsfeld mit mehreren Optionen zu definieren. Dieses Schlüsselwort wird im folgenden Format definiert:

SNGCHCFLD(*RSTCSR/*NORSTCSR, *AUTOSLT/ *NOAUTOSLT/*AUTOSLTENH, *SLTIND/*NOSLTIND, *AUTOENT/*NOAUTOENT/ *AUTOENTENH, *NUMCOL # der Spalten, *NUMROW # der Zeilen, *GUTTER Bundstegweite)

Die ersten drei Parameter entsprechen den Parametern für das Schlüsselwort SFLSNGCHC. Der nächste Parameter wird verwendet, um anzugeben, ob der Datensatz sofort bei Auswahl zurückgegeben werden soll. Die Option *AUTOENT definiert, dass der Datensatz zurückgegeben werden soll, sobald eine Auswahl erfolgt ist. *NOAUTOENT wird angegeben, wenn Sie möchten, dass der Benutzer die Eingabetaste oder eine Befehlstaste drückt, um den Datensatz zurückzugeben. *AUTOENTNN wird verwendet, um festzulegen, dass der Bildschirm automatisch zurückgegeben werden soll, wenn der auf der Workstation verwendete Bildschirm erweiterte Anzeigefunktionen unterstützt.

Die nächsten drei Parameter definieren das Layout der Optionen. Der Parameter *NUMCOL legt die Anzahl der vertikalen Spalten fest. *NUMROW definiert die Anzahl der Zeilen, die für die Anzeige der Optionen verwendet werden. Der letzte Parameter *GUTTER informiert über die Anzahl der Leerstellen, die zwischen den Optionsspalten eingefügt werden sollten. Dieses Schlüsselwort wird im Pulldown-Datensatz selbst definiert, es muss kein zusätzlicher Datensatz definiert werden wie bei den Subfile-Datensätzen. Die Verwendung erfolgt in Verbindung mit Schlüsselwort CHOICE, mit dem die einzelnen Optionen festgelegt werden.

Das Schlüsselwort CHOICE hat folgendes Format:

```
CHOICE( Option #, Optionstext, *SPACEB)
```

Der erste Parameter ist erforderlich und wird verwendet, um den Wert festzulegen, der an das Feld zurückgegeben werden wird, das das Schlüsselwort SNGCHCFLD enthält. Der zweite Parameter, der ebenfalls erforderlich ist, wird verwendet, um den Text

festzulegen, der für die Option angezeigt werden soll. Innerhalb des Felds kann eine Tastenkombination definiert werden, auf die gleiche Weise wie im Feld für das Schlüsselwort MNUBARCHC, wobei das Größer-als-Zeichen verwendet wird. Der letzte Parameter (*SPACE) wird verwendet, um eine leere Zeile zwischen den Optionen einzufügen, deren Optionsnummern zusammenhängend vergeben werden.

Wenn Optionen nicht zusammenhängend sind und in vertikalen Spalten angezeigt werden, wird automatisch ein Leerraum eingefügt. Hiermit können die Optionen auf die gleiche Weise logisch gruppiert werden wie Microsoft Word-Gruppen, bei denen die Optionen „Neu“, „Öffnen“ und „Schließen“ im Dateimenü separat von den Optionen „Seite einrichten“, „Drucken“ und „Druckvorschau“ angeordnet sind. Mit dieser Technik erhält Ihre Anwendung ein ausgereifteres Design. Abbildung 2.9 zeigt Ihnen, wie Sie das Schlüsselwort SNGCHCFLD verwenden.

```
  .
  .
A      R FILE                         PULLDOWN(*RSTCSR *NOSLTIND)
A        OPTION         2Y 0B  1  2SNGCHCFLD
A                                     CHOICE(1 '>New ')
A                                     CHOICE(2 '>Open ')

A                                     CHOICE(3 '>Close ')
A                                     CHOICE(5 '>Save ')
A                                     CHOICE(6 'Save >As')
  .
  .
```

Abbildung 2.9: Das Schlüsselwort SNGCHCFLD für einen PULLDOWN-Datensatz.

In diesem Beispiel verfügt der Pulldown-Datensatz über keinen verknüpften Subfile-Datensatz, sondern das Menü wird vollständig innerhalb des Pulldown-Datensatzes definiert. Das Schlüsselwort SNGCHCFLD ist mit fünf CHOICE-Schlüsselwörtern verknüpft. Wir übergehen Option Nummer 4, um einen Leerraum zwischen den Optionen 3 und 5 einzufügen. Wie oben beschrieben, wird hiermit eine logische Gruppierung ähnlicher Optionen erreicht. Abbildung 2.10 ist ein Code-Abschnitt des ILE-RPG-Codes, der für die Verarbeitung des Menüs verwendet wird.

```
  .
  .
C                   DoW         *INKC=*OFF
C                   ExFmt       BAR1
C                   If          *INKC=*OFF
C                   Select
C                   When        MNUFLD=1
C                   Read        File
C                   If          %EOF(MNU002DS)=*Off
C                   Select
C                   When        Option=1
C                   Exsr        New
C                   EndIf
C                   When        Option=2
C                   Exsr        Open
C                   EndIf
C                   When        Option=3
C                   Exsr        Close
C                   EndIf
C                   When        Option=5
C                   Exsr        Save
C                   EndIf
C                   When        Option=6
C                   Exsr        SaveAs
C                   EndIf
  .
  .
```

Abbildung 2.10: Dieser ILE RPG Code wird verwendet, um den PULLDOWN-Datensatz zu verarbeiten.

Wir ersetzen den Befehl READC, der verwendet wird, um den Subfile-Pulldown-Datensatz in MNU001RG mit einem einfachen READ-Befehl für den Display-File-Datensatz zu verarbeiten, der das Schlüsselwort PULLDOWN enthält. Wir sind auch dazu in der Lage, die gewählte Option zu bestimmen, indem wir die Optionsnummer anstatt den Optionstext verwenden. Dieses Beispiel erfordert weniger Code als die Erstellung des Pulldown-Menüs mit Hilfe einer Subfile. Die Subfile-Methode erlaubt es Ihnen jedoch, die Anzahl der Optionen zu ändern, die im Menü „Ohne Umweg“ von Ihrer Anwendung aus angezeigt werden. Das ermöglicht Ihnen, Menüoptionen gegebenenfalls zu entfernen oder hinzuzufügen. Wenn Sie das Pulldown-Menü erstellen und dabei das Schlüsselwort MLTCHCFLD verwenden, können Sie ein Pulldown-Menü anzeigen, bei dem mehrere Optionen ausgewählt sind. Dieses Schlüsselwort verwendet das folgende Format:

```
MLTCHCFLD( *RSTCSR/*NORSTCSR, *SLTIND/*NOSLTIND, *NUMCOL # der Spalten,
                *NUMROW # der Zeilen, *GUTTER Bundstegweite)
```

Damit können Sie ein Pulldown-Menü erstellen, dessen Menüoptionen aktiviert oder deaktiviert sind. Wie beim Schlüsselwort SNGCHCFLD wird das Schlüsselwort CHOICE verwendet, um die Optionen für das Feld mit Mehrfachauswahl zu definieren. Zusätzlich wird das Schlüsselwort Choice Control (CHCCTL) verwendet, um sowohl die Verfügbarkeit einer Option zu steuern als auch um zu bestimmen, ob eine Option ausgewählt ist. Das Schlüsselwort CHCCTL wird wie folgt angegeben:

```
CHCCTL(Option #, &ctl Feldname, Nachrichten # Library/Datei)
```

Die beiden ersten Parameter sind erforderlich. Die Optionsnummer verknüpft das Schlüsselwort CHCCTL mit dem Schlüsselwort CHOICE. Der zweite Parameter definiert den Namen des verborgenen Felds, über das die Steueroptionen für dieses Feld übergeben werden. Diese Optionen werden auf die gleiche Weise wie das Schlüsselwort SFLCHCCTL eingesetzt. Die Liste der Werte für dieses Feld wird in Tabelle 2.2 angezeigt.

Die letzten drei Parameter sind optional und werden angegeben, um Nachrichten für die Anzeige zu definieren, wenn eine Option, die nicht verfügbar ist, ausgewählt wird. Diese Parameter können entweder als Literal definiert werden, das innerhalb der Display File selbst programmiert wird, oder als der Namen eines Program-to-System-Felds, der verwendet wird, um die Werte zwischen der Anwendung und der Display File zu übergeben. Abbildung 2.11 zeigt ein Beispiel für einen Pulldown-Datensatz, der mit dem Schlüsselwort MLTCHCFLD erstellt wird.

```
  .
  .
A          R OPTIONS                     PULLDOWN(*RSTCSR *SLTIND)
A            OPTION          2Y 0B  1  2MLTCHCFLD
A                                      CHOICE(1 '>Rechtschreibung')
A                                      CHOICE(2 '>Grammatik')
A                                      CHOICE(3 '>Auto. Speichern')
A                                      CHCCTL(1 &CHC1)
A                                      CHCCTL(2 &CHC2)
A                                      CHCCTL(3 &CHC3)
A            CHC1            1Y 0H
A            CHC2            1Y 0H
A            CHC3            1Y 0H
  .
  .
```

Abbildung 2.11: Ein PULLDOWN-Datensatz mit dem Schlüsselwort MLTCHCFL.

In diesem Beispiel wird ein Pulldown-Menü mit dem Namen „Options“ erstellt, das drei Optionen enthält: Rechtschreibung, Grammatik und Auto-Speichern. Jede dieser Optionen fungiert als „Ein-/Aus-Schalter“, der verwendet wird, um die Funktion zu aktivieren oder zu deaktivieren. Der Parameter SLTIND, der im Schlüsselwort PULLDOWN angegeben wird, führt dazu, dass die Optionen mit einem Häkchen links vom Optionstext angezeigt werden. Das macht es einfacher, festzustellen, ob eine Option ausgewählt ist oder nicht.

Um den Status der Optionen in einem Mehrfachauswahl-Feld zu bestimmen (d.h. aktiviert oder nicht aktiviert) initiieren wir eine READ-Operation im Pulldown-Datensatz, der das Mehrfachauswahlfeld enthält, auf die gleiche Weise wie beim Einfachauswahl-Feld. Als Nächstes prüfen wir den Wert der Variablen CHC1, CHC2, und CHC3. Für eine aktivierte Option hat die Variable CHCTL den Wert 1. Für Optionen, die nicht aktiviert sind, wird die Variable CHCTL auf 0 gesetzt. Ihre Anwendung kann dann den Wert dieser Variablen ähnlich wie einen *IN-Indikator in RPG III verwenden. In unserem Beispiel können wir den Status von CHC1 prüfen, bevor wir die Rechtschreibprüfungs-Subroutine aufrufen, wie in Abbildung 2.12 gezeigt.

```
  .
  .
C                     DoW       *INKC=*OFF
C                     ExFmt     BAR1
C                     If        *INKC=*OFF
C                     Select
C                     When      MNUFLD=4
C                     Read      Options
  .
  . zusätzlicher Code
  .
C                     If        CHC1=1
C                     Exsr      SpellCheck
C                     EndIf
  .
  .
```

Abbildung 2.12: Dieser Code verarbeitet ein Mehrfachauswahlfeld (MLTCHCFLD).

Wie Sie sehen können, lässt die Verwendung von Pulldown-Menüs Ihre Anwendungen nicht nur besser aussehen, sie hilft auch dabei, den Programmfluss logischer, unkomplizierter und leichter analysierbar zu gestalten.

Zusätzliche GUI-Objekte

Bei der Besprechung von Pulldown-Menüs haben wir uns oberflächlich mit einigen anderen erweiterten Anzeigefunktionen befasst, wie z.B. mit Feldern für die Einfach- und Mehrfachauswahl („Single/Multiple Choice“) sowie Subfile-Feldern für die Mehrfachauswahl. Diese Objekte können außerhalb der Pulldown-Menüs verwendet werden, um weitere Ergebnisse zu erzielen. Ein Feld, das mit dem Schlüsselwort SNGCHCFLD geprüft wird, kann als Optionsschaltfläche dienen, mit der eine Option aus einer Liste von Optionen in einem Standarddatensatz oder innerhalb eines Fensters gewählt werden kann.

Ein Beispiel für einen sinnvollen Einsatz dieses Objekttyps ist die Verwendung in einem Bedienerführungsbildschirm für ein Berichtsprogramm, das den Masterdatensatz auflistet, der es dem Benutzer erlaubt, entweder nur Parent-Elemente, Komponenten-Elemente oder alle Elemente auszuwählen. In einer Standard-Display-File würden Sie eventuell ein Zeichenfeld mit einer einzigen Position erstellen und den Benutzer auffordern, P, C oder A einzugeben, und danach das DDS-Schlüsselwort VALUES (‘P’ ‘C’ ‘A’) der Display File-Definition des Felds hinzufügen.

Obwohl diese Technik funktionieren würde, muss der Benutzer die Option (P, C oder A) eingeben und die Eingabetaste drücken. Außerdem erfolgt erst eine Prüfung, nachdem die Eingabetaste gedrückt wurde und an den Benutzer eine Fehlernachricht zurückgegeben wird. Wir können dieses Feld mit einem einstelligen numerischen Feld ersetzen, das die Schlüsselwörter SNGCHCFLD und CHOICE enthält, wie in Abbildung 2.13 gezeigt.

```
A          R RADIO
A                                  6  4'ITEMS:'
A            SELECT         2Y 0B  6 12SNGCHCFLD
A                                      CHOICE(1 'PARENT ITEMS ONLY')
A                                      CHOICE(2 'CHILD ITEMS ONLY')
A                                      CHOICE(3 'ALL ITEMS')
```

Abbildung 2.13: Ein Display-File-Datensatz mit Schlüsselwort SNGCHCFLD.

Das Resultat ist, dass der Benutzer einfach die gewünschte Option mit Hilfe der Pfeiltasten oder der Maus markiert, und die Option wird ausgewählt. Eine Prüfung muss nicht erfolgen, weil die Werte strikt auf die Werte beschränkt sind, die im Schlüsselwort CHOICE definiert sind, ganz zu schweigen davon, dass diese Technik einfach viel besser aussieht. Abbildung 2.14 zeigt eine Gegenüberstellung der beiden Techniken.

Abbildung 2.14: Die Technik SNGCHCFLD im Vergleich mit einem Standardeingabefeld.

Das Schlüsselwort MLTCHCFLD kann auf ähnliche Weise verwendet werden, um eine Gruppe von Kontrollkästchen zu erstellen, die z.B. verwendet werden können, um Felder für die Anzeige in unserem Bericht auszuwählen.

Abbildung 2.15 zeigt ein Beispiel für einen Display-File-Datensatz für genau diesen Zweck. Der Benutzer klickt auf den Feldnamen für jedes Feld, das im Bericht erscheinen soll. Anschließend wird das Kontrollkästchen aktiviert und der Wert des Steuerfelds CHCCTL lautet 1.

```
A          R CHKBOX
A                                    7 10'SELECT FIELDS FOR REPORT:'
A            CHECK          2Y 0B  8 10MLTCHCFLD
A                                       CHOICE(1 'ITEM NUMBER')
A                                       CHCCTL(1 &O1)
A                                       CHOICE(2 'ITEM DESCRIPTION')
A                                       CHCCTL(2 &O2)
A                                       CHOICE(3 'BRAND')
A                                       CHCCTL(3 &O3)
A                                       CHOICE(4 'ON HAND QTY')
A                                       CHCCTL(4 &O4)
A                                       CHOICE(5 'ON ORDER QTY')
A                                       CHCCTL(5 &O5)
A                                       CHOICE(6 'Y-T-D SALES $')
A                                       CHCCTL(6 &O6)
A            O1        1Y 0H
A            O2        1Y 0H
A            O3        1Y 0H
A            O4        1Y 0H
A            O5        1Y 0H
A            O6        1Y 0H
```

Abbildung 2.15: Das Schlüsselwort MLTCHCFLD kann für die Anzeige von Kontrollkästchen verwendet werden.

Alle nicht aktivierten Felder geben 0 zurück. Wie Sie sehen, ist es ganz einfach, zu bestimmen, welche Optionen ausgewählt sind. Und vergessen Sie nicht, Sie müssen sich keine Gedanken darüber machen, ob der Anwender eine gültige Option in ein Auswahlfeld eingibt. Abbildung 2.16 zeigt, wie ein Feld mit MLTCHCFLD angezeigt wird.

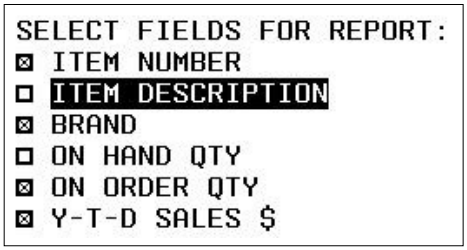


Abbildung 2.16: Ein Display-File-Datensatz unter Verwendung von MLTCHCFLD.

Die beiden Subfile-Schlüsselwörter, die wir in den vorherigen Beispielen verwendet haben, können in einem Standard-Subfile-Bildschirm völlig anders aussehen; sogar noch besser sehen sie aus, wenn sie mit einem Fenster kombiniert werden. Das Endresultat ist einem Listenfeld sehr ähnlich. Denken Sie daran, dass die Schlüsselwörter SFLSNGCHC oder SFLMLTCHC beide in Subfile-Steuerdatensätzen für Subfiles mit einem einzelnen Ausgabefeld verwendet werden. Damit erhalten Sie die Möglichkeit, eine Liste zu erstellen und es dem Benutzer zu erlauben, ein Element aus der Liste auszuwählen, indem er einfach darauf klickt oder die Pfeiltasten und die Eingabetaste zuhilfe nimmt.

Die Display File LST002DS ist ein Beispielfenster, das das Schlüsselwort SFLSNGCHC verwendet. Das ILE RPG-Programm LST002RG wird verwendet, um diesen Bildschirm zu steuern. Dieses Programm ermöglicht es einem Benutzer, einen U.S.-Staat auszuwählen. Obwohl das ein einfaches Beispiel ist, kann es als Grundlage für alle Programme herhalten, die diese Technik verwenden müssen. Der Subfile-Datensatz SFLSNGCHC kann zwar nur ein Ausgabefeld enthalten, aber Sie können diesen Feldwert entweder mit anderen Werten oder durch Verwendung einer Datenstruktur benutzen, um die Felder zu kombinieren. Der Vorteil an der Verwendung einer Datenstruktur für diesen Zweck ist es, dass Sie beliebige Felder aus den Listenfeldern problemlos einlesen können, indem Sie das Listenfeld in die Datenstruktur verschieben. Diese

Technik wird in unserem Beispiel eingesetzt, um uns die Anzeige von der Abkürzung, die aus zwei Buchstaben besteht, und der Staatsbezeichnung zu ermöglichen. Wenn dann der Benutzer eine Auswahl trifft, wird nur der Wert des Abkürzungsfelds von unserer Anwendung verwendet; der Wert beliebiger Felder in der Datenstruktur kann jedoch ebenfalls abgerufen werden. Den Unterschied im Vergleich zum Erscheinungsbild der ursprünglichen Version dieses Programms, das in Abbildung 2.6 gezeigt wurde, sehen Sie in Abbildung 2.17. Wie Sie sehen können, wurde das Auswahlfeld von der linken Seite des Fensters entfernt. Optionen werden nun mit Hilfe der Pfeiltasten oder dem Mauszeiger markiert.

Das Schlüsselwort SFLMLTCHC kann auf ähnliche Weise verwendet werden, um ein Listenfeld zu erstellen, das es dem Benutzer ermöglicht, mehrere Optionen zu wählen. Um festzulegen, welche Option oder Optionen für einen Datensatz unter Verwendung des Schlüsselworts SFLMLTCHC ausgewählt wurden, müssen wir den Wert des Felds prüfen, der das Schlüsselwort SFLCHCTL enthält. Wie bereits beschrieben, enthält dieses Feld 0, wenn der Datensatz nicht ausgewählt ist, und 1, wenn er ausgewählt ist. Aber die Schlüsselwörter SNGCHCFLD, MLTCHCFLD, SFLMLTCHC und SFLSNGCHC sind nicht die einzigen erweiterten Anzeigeschlüsselwörter, die wir verwenden können, um unsere Anzeige mit Funktionalität und Stil zu ergänzen.

Das Schlüsselwort Push Button Field (PSHBTNFLD) wird für ein numerisches Feld definiert, das die Länge 2 Stellen mit 0 Dezimalstellen aufweist, um eine Reihe von Schaltflächen-Controls zu erstellen. Schaltflächen können verwendet werden, um den Fluss der Programmlogik zu steuern, weil die Steuerung an die Anwendung zurückgegeben wird, sobald auf eine Schaltfläche geklickt wird;

der entsprechende Wert der Schaltfläche wird zurückgegeben, sobald der Feldwert für das Feld das Schlüsselwort PSHBTLFLD enthält.

Dieses Schlüsselwort weist folgendes Format auf:

```
PSHBTNFLD(*RSTCSR/*NORSTCSR , *NUMCOL, *NUMROW, *GUTTER)
```

Alle Parameter sind optional. Der Parameter *RSTCSTR/*NORSTCSR wird verwendet, um den Benutzer daran zu hindern, den Cursor außerhalb des Schaltflächenfelds zu verschieben. Der optionale Parameter *NUMCOL definiert die Anzahl der Spalten, in denen die Schaltflächen angezeigt werden sollen. Der optionale Parameter NUMROW wird angegeben, um die Anzahl der Zeilen festzulegen, die bei der Anzeige Ihrer Schaltflächen verwendet werden sollen. Das letzte Schlüsselwort *GUTTER definiert die Anzahl der Leerstellen, die zwischen den Spalten der Schaltflächen eingefügt werden sollen. Wenn der Parameter *GUTTER im Schlüsselwort PSHBTNCTL nicht angegeben wird, wird ein Standardwert von drei Leerstellen verwendet. Eine Zahl, die größer ist als Null und die es ermöglicht, dass alle Schaltflächen in den Bildschirm passen, würde diesen Parametern folgen.

Ein Feld, das das Schlüsselwort Push Button Field (PSHBTNFLD) enthält, muss auch das Schlüsselwort Push Button Choice (PSHBTNCHC) enthalten, das verwendet wird, um die Schaltflächen innerhalb von PSHBTNCTL zu definieren. Dieses Schlüsselwort wird auf Feldebene definiert und in diesem Format verwendet:

```
PSHBTNCHC(Schaltfläche #, Schaltflächentext, cmd-Taste *SPACEB)
```

Der Parameter **Schaltfläche #** wird angegeben, um zu bestimmen, auf welche Schaltfläche geklickt wurde. Dieser Parameter sollte als Ganzzahl zwischen 1 und 99 angegeben werden. Der Parameter **Schaltflächentext** definiert den Text, der auf der Schaltfläche angezeigt werden soll. Wie bereits beschrieben, kann auch das Größer-als-Zeichen (>) verwendet werden, um eine Tastenkombination für eine Schaltfläche zu definieren. Der Parameter **cmd-Taste** wird festgelegt, um eine Befehlstaste mit der Schaltfläche zu verknüpfen. Wenn die Befehlstaste gedrückt wird, wird das Drücken der Schaltfläche stimuliert. Der Parameter *SPACEB kann angegeben werden, um eine leere Zeile zwischen der definierten Schaltfläche und der vorherigen Schaltfläche einzufügen. Die DDS-Display File Source in Abbildung 2.18 zeigt, wie Schaltflächenfelder verwendet werden sollten.

```
A          R AREUSURE                  WINDOW(&X &Y 4 25)
A                                      WDWTITLE((*TEXT 'SIND SIE SICHER'))
A            ANSWER         2Y 0B  2 3PSHBTNFLD
A                                       PSHBTNCHC(1 '>Ja')
A                                       PSHBTNCHC(2 '>NEIN' CA12)
A                                       CHCCTL(1 &CHC1)
A                                       CHCCTL(2 &CHC2)
A            CHC1           1Y 0H
A            CHC2           1Y 0H
A            Y  2S 0P
A            X  3S 0P
```

Abbildung 2.18: Das „Sind-Sie-sicher“-Fenster verwendet ein Schaltflächenfeld.

Mit diesem Beispiel wird ein einfaches „Sind-Sie-sicher“-Fenster erstellt, das zwei Schaltflächen enthält. Eine Schaltfläche hat den Titel „Ja“, die andere die Bezeichnung „Nein.“ Wenn dieses Fenster vom Typ Dialog-Feld angezeigt wird, hat der Benutzer die

Möglichkeit, auf eine dieser Schaltflächen zu klicken. Wenn der Benutzer auf die Schaltfläche Ja klickt, wird der Wert 1 in das Feld ANSWER zurückgegeben. Wenn Nein aktiviert wird, wird der Wert 2 zurückgegeben. Das Schlüsselwort CHCCTL wird hier erneut verwendet, um die Verfügbarkeit einer bestimmten Schaltfläche zu kontrollieren, indem eine der zuvor beschriebenen Werte für dieses Schlüsselwort angegeben wird.

Sie werden auch bemerken, dass wir die Positionswerte im Schlüsselwort WINDOW mit den beiden Program-to-System-Feldern ersetzt haben (&X und &Y), die es uns ermöglichen, die Position des Fensters von unserer Anwendung aus festzulegen, wenn das Fenster ausgegeben wird. Diese Funktion ist ideal geeignet, wenn das Fenster oberhalb eines anderen Bildschirms angezeigt wird und sich auf ein bestimmtes Feld oder einen bestimmten Bereich des Bildschirms bezieht. Ein Beispiel wäre eine Anwendung, in der der Benutzer eine Befehlstaste drückt, um anzugeben, dass er den Wert in einem bestimmten Feld aktualisieren möchte.

Nach der Aktualisierung kann dieses Fenster verwendet werden, um zu prüfen, ob der Benutzer eine Aktualisierung wünscht, bevor die Anwendung die Datei aktualisiert. Dieser Code erstellt ein Control, das einem Dialogfeld sehr ähnlich ist. Die Schaltflächen selbst werden aktiviert, indem der Benutzer mit der Tabulatortaste sowie mit den Pfeiltasten zur Schaltfläche wechselt und diese mit der Leertaste auswählt, oder durch einen einfachen Klick auf diese Schaltfläche mit einem Mauszeiger.

Dies ist ein einfaches, aber effektives Beispiel für die Einsatzmöglichkeiten der erweiterten Anzeigefunktionen.

Die Ausgabe dieses Objekts sieht ähnlich wie die Abbildung 2.19 aus.

Abbildung 2.19:
Ein erstelltes Dialogfeld

Der Benutzer kann zur gewünschten Option entweder mit den Pfeiltasten oder mit dem Mauszeiger wechseln. Dieses Steuerelement ändert nicht nur das Aussehen Ihrer Anwendung, sondern verändert von Grund auf den Fluss des Programms.

Denken Sie jedoch daran, dass diese zusätzliche Funktionalität nicht mit dem Anwendungsfluss in Konflikt kommen sollte. Vergessen Sie nicht, dass Ihr Ziel nicht sein sollte, die Anwendung auf Kosten der Anwenderfreundlichkeit zu verschönern. In unserem „Sind-Sie-sicher"-Beispiel werden eine Funktionstaste und eine Meldung („Drücken Sie F8, um die Aktualisierung zu bestätigen") mit einem Fenster ersetzt. Wenn Sie diese Funktionalität nur deshalb hinzufügen, um das Aussehen der Anwendung zu ändern, kann die Benutzerfreundlichkeit des Programms leiden. Das Hinzufügen von „Sind Sie sicher" zu diesem Beispiel hätte in diesem Fall keinen Sinn. Das Beispiel soll Ihnen lediglich die Verwendung der neuen Steuerelemente zeigen.

Versuchen Sie, das Anwendungsdesign nicht unnötig kompliziert zu machen, wenn Sie neue Funktionen hinzufügen. Eine gute Technik, die ich vorschlagen würde, ist der Versuch, bei jeder Gelegenheit das „Look and Feel" von Microsoft Windows-Anwendungen nachzuahmen. Da die meisten Anwender es gewohnt sind, mit Windows-Programme zu arbeiten, wird für sie die Verwen-

dung Ihrer Anwendung komfortabler, je mehr Ihr Programm wie eine Windows-Anwendung aussieht. Ob Sie erweiterte Anzeigefunktionen verwenden oder nicht, es ist immer eine gute Regel, das „Look and Feel“ Ihrer Anwendungen einheitlich zu gestalten. Dies kann gemeinsam mit den erweiterten Anzeigefunktionen dabei helfen, Ihre Anwendungen intuitiver und benutzerfreundlicher zu gestalten.

3

Benutzerdefinierte ILE RPG-Funktionen

Eine Methode, die Ihnen dabei hilft, sämtliche Anwendungsprozesse einheitlich zu gestalten, besteht darin, diese „modular“ zu entwickeln. Das heißt, Sie verwalten gemeinsame Funktionen zentral und rufen sie von den einzelnen Anwendungen aus auf, die diese Funktionalität benötigen. An diesem Punkt kommen die benutzerdefinierten Funktionen von ILE RPG ins Spiel. Wenn Sie jemals eine Anwendung in Visual Basic, C++, Java oder in einer anderen modernen Programmiersprache geschrieben haben, wissen Sie, dass es die Arbeit sehr erleichtert, wenn Sie eine Programmaufgabe, die generell verwendet wird, mit Hilfe einer Funktion automatisieren können. Denn so ist es nicht mehr erforderlich, Dutzende Programme zu modifizieren, wenn der Code, von dem eine Aufgabe ausgeführt wird, aktualisiert werden muss – z.B. bei der Behebung eines Fehlers. Mit ILE RPG können Sie auch benutzerdefinierte Funktionen erstellen, die ohne großen Aufwand von jeder ILE-Anwendung aus verwendet werden können. Angenommen, Sie haben eine physische Datei erstellt, die einen Kundennamen und eine Adresse enthält. In diesem Fall kann im Handumdrehen eine benutzerdefinierte Funktion entwickelt werden, in die die Kunden-ID-Nummer eingegeben wird, und die anschließend den Kundennamen zurückgibt.

Wie Sie sich sicherlich erinnern, kann eine benutzerdefinierte Funktion in ILE auf die gleiche Weise aufgerufen werden wie die integrierten ILE-Funktionen, indem der Opcode EVAL wie folgt aufgerufen wird:

```
C Eval X=Function(Parm:Parm)
```

Stellen Sie sich vor, Sie könnten ganze Subroutinen in Ihren Anwendungen mit einer Handvoll benutzerdefinierten Funktionen

ersetzen. Bei jeder Änderung müssten Sie diese nur einmal kodieren und testen. Überlegen Sie sich, wie viele Anwendungen, mit denen Sie momentan arbeiten, die zuvor beschriebenen Funktionen beherrschen: Einlesen eines Kundennamens aus einer Kunden-Masterdatei.

Und jetzt stellen Sie sich vor, dass Sie den gesamten Code, der für das Einlesen dieser Informationen erforderlich ist, mit einem einfachen EVAL-Befehl ersetzen:

```
C Eval CusName=ReadCust(CustID)
```

Der Vorteil davon ist, dass Sie nicht nur die CHAIN-Funktionalität nutzen können, die in die benutzerdefinierte Funktion integriert ist, sondern Sie können außerdem integrierte Fehlerbehandlungsfunktionen einsetzen. Sie könnten z.B. Code *ERROR zurückgeben, wenn ein Kunde nicht dem angegebenen Wert im Feld CustID entspricht, anderenfalls wird der Kundenname zurückgegeben. Wenn wir nun aus irgendeinem Grund den Code modifizieren müssten, um zum Beispiel zu prüfen, ob das Kundenkonto inaktiv ist, müssten wir nur ein einziges Programm verändern. Sehen wir uns an, wie eine benutzerdefinierte Funktion in ILE RPG in der Praxis erstellt wird.

Programmierung der Module

Funktionen können in ILE-RPG mit zwei Methoden erstellt werden. Die erste besteht darin, die Source für das Modul, die die Funktion enthält, direkt in Ihr Anwendungsprogramm einzufügen. Mit dieser Methode wird der gesamte Code nach dem Anwendungscode eingefügt, davon ausgenommen der Funktionsprototyp. *Prototypen* werden verwendet, um die benutzerdefinierte Funktion für Ihr Programm zu deklarieren. Die zweite Methode erfordert, dass eine oder mehrere Funktionen in ein ILE RPG-Modul eingefügt werden. Bei dieser Methode muss anschließend das aufrufende Programm mit dem Prototyp ergänzt werden. Der Prototyp definiert die Parameter, die in Ihre Funktion gesendet werden, sowie den Typ „Wert“, der zurückgegeben werden wird.

Prototypen werden in den D-Spezifikationen definiert und bestehen aus dem Prototypdatensatz selbst und seinen Parameterdefinitionen. Der Prototypdatensatz, der in den Positionen 24 und 25 mit PR markiert wurde, legt den Funktionsprozedurnamen fest und stellt eine Typdefinition des Werts zur Verfügung, der zurückgegeben werden soll. Auf den Prototypdatensatz folgen die Parameterdatensätze, von denen der Typ der Daten definiert wird, die in die Funktion oder Prozedur übergeben werden soll. In ILE RPG werden die Begriffe *Prozedur, Subprozedur* und *Funktion* häufig austauschbar verwendet.

Die einzelnen Begriffe haben jedoch unterschiedliche Bedeutungen. Jedes ILE RPG-Programm enthält z.B. eine Prozedur. Ihr Programm selbst ist eine Prozedur – mit der Bezeichnung *Main Procedure* (Hauptprozedur). Alle zusätzlichen Prototypen, die innerhalb der Programm-Source definiert sind, werden als Subproze-

duren betrachtet. Diese Subprozeduren können nicht selbstständig ausgeführt werden. Eine Funktion wiederum ist eine Subprozedur, die einen Wert an das aufrufende Programm zurückgibt – wie zuvor mit Hilfe des Befehls EVAL gezeigt. Subprozeduren, die keinen Wert zurückgeben, können aus der Main Procedure oder von einer anderen Subprozedur aus ausgeführt werden, indem der Opcode Call Bound Procedure (CALLB) verwendet wird. Der in Abbildung 3.1 gezeigte Source-Code zeigt ein Beispiel für den Prototyp für die ReadCust-Funktion, die wir bereits besprochen haben.

```
*----------------------------------------------------------------------*
* ReadCust Function Prototype
*----------------------------------------------------------------------*
 D ReadCust         PR          30I 0
 D  CustID                       8    VALUE
 D  ShipToID                     3    VALUE OPTIONS(*NOPASS)

 D  RtnLength                    5  0 VALUE OPTIONS(*NOPASS)
```

Abbildung 3.1: Dieser Prototyp definiert die benutzerdefinierte ReadCust-Funktion.

Es wird Ihnen sicherlich auffallen, dass mehr Parameter definiert wurden, als wir im Beispiel zuvor verwendet haben. Parameter, die OPTION(*NOPASS) enthalten, sind optional und können beim Aufruf der Funktion ausgeschlossen werden. Wie bereits erwähnt, können Sie auch ein Modul erstellen, das mehrere Funktionen für den allgemeinen Einsatz oder für Subprozeduren enthält. Dieser Modultyp wird als optionales NOMAIN-Modul betrachtet, weil es keine Main Procedure enthält.

Wenn benutzerdefinierte Funktionen und Subprozeduren verwendet werden, wird normalerweise die Source für den Prototyp oder

die Prototypen in eine eigene Quellenteildatei eingefügt und in Ihre Anwendung und das NOMAIN-Modul integriert, falls dieses Modul verwendet wird. Dabei wird die Compiler-Direktive /COPY verwendet. Damit kann die Anzahl der Änderungen an der Anwendung selbst stark reduziert werden, weil diese Source nur einmal modifiziert werden muss. Ein NOMAIN-Modul wird erstellt, indem die NOMAIN-Direktive in die H-Spezifikation des Moduls eingefügt wird. Damit wird der Compiler informiert, dass dieses Programm keine Main Procedure besitzt. Weil Subprozeduren selbst nicht ausgeführt werden können, kann der Aufruf nicht von einer Befehlszeile aus erfolgen.

Die Antwort auf die Frage, ob Sie Ihre Funktionen innerhalb der Anwendung oder extern innerhalb eines NOMAIN-Moduls erstellen sollen, hängt von mehreren Punkten ab. Erstens von der Art der Funktion: Kann sie von mehreren Anwendungen in Ihrem gesamten System genutzt werden, oder wird diese Funktion nur in der gesamten Anwendung benötigt? Eine Funktion zum Beispiel, mit der berechnet wird, ob ein bestimmtes Datum ein Arbeitstag ist, könnte in vielen Anwendungen nützlich sein. Dem gegenüber könnten Sie eine Funktion verwenden, die den Skontobetrag für einen Kunden berechnet und die innerhalb Ihrer Auftragserfassungsanwendung mehrmals aufgerufen wird, für die sich jedoch außerhalb dieser Anwendung wenig Einsatzmöglichkeiten bieten.

Maßgeblich bei Ihrer Entscheidung, wo Sie Ihre Funktion positionieren sollten, ist die Frage, ob die Funktion so allgemein gehalten werden kann, dass die Integration in andere Anwendungen möglich ist. In der zuvor besprochenen Funktion ReadCust wurde diese Eigenschaft sichergestellt, indem die Angabe von zwei optionalen Parametern ermöglicht wurde, die es uns erlaubt, eine

„Ship-to-ID“ sowie die Länge des zurückgegebenen Namens zu definieren.

```
*----------------------------------------------------------------------*
* DateText Function Prototype
*----------------------------------------------------------------------*
 D DateText         PR                  35I  0
 D  DateValue                            7S  0VALUE
```

Abbildung 3.2: Dieser Prototyp für die Funktion DateText benötigt ein siebenstelliges Datum (cyymmdd).

Damit können Sie auch die Art und Weise beeinflussen, in der Sie die Funktion selbst entwickeln. D.h., wenn Sie erkennen, dass eine Funktion in mehreren Anwendungen verwendet werden kann, sollten Sie in Betracht ziehen, die Funktion selbst so zu entwickeln, dass der flexible Einsatz möglich ist. In einigen Fällen heißt das, dass Sie optionale Parameter hinzufügen. Ein Beispiel dafür wäre die Erstellung einer Funktion für die Konvertierung eines Datumsfelds in eine Textdarstellung des Datums (Monatsname, dd, yyyy) mit Hilfe des in Abbildung 3.2 gezeigten Prototyps. Wenn die Funktion anfangs für eine bestimmte Anwendung entwickelt wurde, die ein siebenstelliges Datum (cyymmdd) einsetzt, können wir davon ausgehen, dass dies der Typ des übergebenen Datums sein wird und die Funktion wie folgt aufrufen:

```
C Eval DateString=DateText(DateVal)
```

Andererseits können wir, um maximale Flexibilität zu gewährleisten, einen optionalen Parameter hinzufügen, wie in Abbildung 3.3 gezeigt, der für die Angabe des Datumsformats verwendet wird.

```
*----------------------------------------------------------------------*
* DateText Funktions-Prototyp
*----------------------------------------------------------------------*
 D DateText         PR                  35A  O
 D  DateValue                           12A   VALUE
 D  DateFmt                              4A   VALUE OPTION(*NOPASS)
```

Abbildung 3.3: Dieser Prototyp für die Funktion DateText akzeptiert einen optionalen Datums-Softwareparameter.

Die Anweisung EVAL, von der die Funktion verwendet wird, wird wie folgt festgelegt:

```
C Eval DateString=DateText(DateVal, '*YMD')
```

Im zweiten Beispiel ändern wir den Parameter DateValue auf einen variablen Datentyp ab. Dies ermöglicht es uns, mit dem angegebenen Parameter den verwendeten Datumstyp zu definieren. Unsere Funktion behandelt anschließend jedes mögliche Datumsformat separat.

Wir verwenden dieses Beispiel, um uns die Erstellung einer Funktion unter Verwendung von ILE RPG anzusehen. Abbildung 3.4 zeigt die Source für das Programm DDT001RG. Denken Sie daran, dass Sie bei der Erstellung von ILE RPG-Funktionen die Möglichkeit haben, die Funktion entweder innerhalb des Programms oder als Subprozedur eines separaten Moduls zu erstellen. In diesem Fall werden wir die Funktion direkt in das RPG-Programm einfügen.

```
H DATFMT(*MDY)
 ****************************************************************************
 *
 * Befehl erstellen: CRTBNDRPG PGM(library/DDT001RG)
 *                             SRCFILE(library/QRPGLESRC)
 *
 ****************************************************************************
 *--------------------------------------------------------------------------*
 *
 * DateText Funktionsprototyp
 *--------------------------------------------------------------------------*
 *
D DateText        PR            35A
D  DateValue                    12A   VALUE
D  DateFmt                       4A   VALUE OPTIONS(*NOPASS)
D strDate         S             35A
D Months          S             10A   CTDATA PERRCD(6) DIM(12)
C*
C                   Movel(P)  UDATE         Date1            12
C                   Eval      strDate=DateText(Date1 : '*MDY')
C     strDate       Dsply
C*
C                   Eval      *INLR=*On
 *--------------------------------------------------------------------------*
 * DateText-Funktion
 *--------------------------------------------------------------------------*
P DateText        B
D DateText        PI            35A
D  DateValue                    12A   VALUE
D  DateFmt                       4A   VALUE OPTIONS(*NOPASS)
D DateFld         S                D   DATFMT(*ISO)
D DateTx          S             35A
D Mth             S              2S 0
D Day             S              2S 0
D Year            S              4S 0
C*
C*  Wenn optionaler Parameter fehlt, Standard verwenden
C                   If %Parms=1
C                   Eval        DateFmt='*MDY'
C                   EndIf
C                   Select
C                   When      DateFmt='*YMD'
C     *YMDO         Movel     DateValue     DateFld
```

Abbildung 3.4: Dieses Programm verwendet die intern definierte Funktion DateText (Teil 1 von 2).

```
 C                   When      DateFmt='*MDY'
 C     *MDY0         Movel     DateValue     DateFld
 C                   When      DateFmt='*CYMD'
 C     *CYMD0        Movel     DateValue     DateFld
 C                   When      DateFmt='*USA'
 C     *USA0         Movel     DateValue     DateFld
 C                   EndSl
 C                   Extrct    DateFld:*MMth
 C                   Extrct    DateFld:*DDay
 C                   Extrct    DateFld:*YYear
 C                   Eval      DateTx=%Trim(Months(Mth)) +
 C                             ', '+ %Trim(%Char(Day))   +
 C                             '  '+ %Trim(%Char(Year))
 C                   Return    DateTx
 P DateText        E
** Monatsnamen
January    February   March      April     May       June
July       August     September  October   November  December
```

Abbildung 3.4: Dieses Programm verwendet die intern definierte Funktion DateText (Teil 2von 2).

In diesem Beispiel wird die integrierte Funktion %PARMS verwendet, um die numerischen Parameter zu bestimmen, die in die Funktion übergeben wurden. Dies erfolgt, um festzustellen, ob optionale Parameter, in diesem Fall das Datumsformat, übergeben wurden. Um dieses Programm zu kompilieren, verwenden Sie den Befehl Create Bound RPG Program (CRTBNDRPG) wie folgt:

```
CRTBNDRPG PGM(library/DDT001RG) SRCFILE(library/QRPGLESRC)
                    DFTACTGRP(*NO) ACTGRP(*CALLER)
```

Die Verwendung des Parameters Default Activation Group (DFTACTGRP) muss als *NO definiert werden, der Parameter Activation Group (ACTGRP) muss als *CALLER definiert werden. Damit geben Sie an, dass sich innerhalb des Programms selbst benutzerdefinierte Funktionen befinden. Die Funktion verschiebt

den übergebenen Wert in ein Feld, das als *ISO-Datumsformat definiert wurde, wobei das Datumsformat verwendet wird, das mit dem Opcode MOVEL definiert wird. Als Nächstes geben wir den Befehl EXTRCT ein, um die Datumsbereiche für Monat, Tag und Jahr zu extrahieren. Die integrierten Funktionen %CHAR und %TRIM werden anschließend mit diesen Werten verwendet, um die numerischen Felder in Text zu konvertieren sowie um führende und abschließende Leerräume aus dem resultierenden Textwert zu entfernen. Wir verwenden außerdem das Kompilierzeit-Array MONTHS für den Abruf der Namen für den erforderlichen Monat.

Als Nächstes untersuchen wir den Prozess der Erstellung der gleichen Funktion in einem eigenen Modul. Als erste und naheliegendste Änderung werden Sie vermutlich dafür sorgen, dass die Funktion selbst in eine separate Quellenteildatei eingefügt wird. Die Abbildungen 3.5 und 3.6 zeigen die überarbeitete Version des Anwendungsprogramms und des Funktionsmoduls.

```
H DATFMT(*MDY)
 ****************************************************************************
 *
 * Befehle erstellen: CRTRPGMOD MODULE(library/DSPDAT)
 *                               SRCFILE(library/QRPGLESRC)
 *
 *                    CRTPGM    PGM(library/DDT002RG) MODULE(DSPDAT DATEFN)
 *                              ENTMOD(*FIRST)
 *
 ****************************************************************************
 *----------------------------------------------------------------------------*
 * DateText-Funktionsprototyp
 *----------------------------------------------------------------------------*
D DateText        PR              35A
D  DateValue                      12A   VALUE
D  DateFmt                         4A   VALUE OPTIONS(*NOPASS)
D strDate         S               35A
```

Abbildung 3.5: Dieses Programm greift auf die Funktion DateText über das Modul DATEFN zu (Teil 1 von 2).

```
C*
C                   Movel(P)   UDATE         Date1            12
C                   Eval       strDate=DateText(Date1 : '*MDY')
C     striate       Dsply
C*
C                   Eval       *INLR=*On
```

Abbildung 3.5: Dieses Programm greift auf die Funktion DateText über das Modul DATEFN zu (Teil 2 von 2).

```
H DATFMT(*MDY) NOMAIN
 ****************************************************************************
 *
 * Befehl erstellen: CRTRPGMOD MODULE(library/DATEFN)
 *                             SRCFILE(library/QRPGLESRC)
 *
 ****************************************************************************
 *----------------------------------------------------------------------------*
 * DateText Funktionsprototyp
 *----------------------------------------------------------------------------*
D DateText        PR            35A
D  DateValue                    12A   VALUE
D  DateFmt                       4A   VALUE OPTIONS(*NOPASS)
D Months          S             10A   CTDATA PERRCD(6) DIM(12)
 *----------------------------------------------------------------------------*
 * DateText Funktion
 *----------------------------------------------------------------------------*
P DateText        B                   EXPORT
D DateText        PI            35A
D  DateValue                    12A   VALUE
D  DateFmt                       4A   VALUE OPTIONS(*NOPASS)
D DateFld         S                D   DATFMT(*ISO)
D DateTx          S             35A
D Mth             S              2S 0
D Day             S              2S 0
D Year            S              4S 0
C*
C*  Wenn optionale Parameter fehlen, Standard verwenden
C                   If        %Parms=1
C                   Eval      DateFmt='*MDY'
C                   EndIf
```

Abbildung 3.6: Diese Version der Funktion DateText verfügt über ein eigenes Modul (Teil 1 von 2).

```
 C                   Select
 C                   When      DateFmt='*YMD'
 C     *YMD0         Movel     DateValue     DateFld
 C                   When      DateFmt='*MDY'
 C     *MDY0         Movel     DateValue  DateFld
 C                   When      DateFmt='*CYMD'
 C     *CYMD0        Movel     DateValue  DateFld
 C                   When      DateFmt='*USA'
 C     *USA0         Movel     DateValue  DateFld
 C                   EndSl
 C                   Extrct    DateFld:*M Mth
 C                   Extrct    DateFld:*D Day
 C                   Extrct    DateFld:*Y Year
 C                   Eval      DateTx=%Trim(Months(Mth)) +
 C                             ', '+ %Trim(%Char(Day))  +
 C                             '  '+ %Trim(%Char(Year))
 C                   Return    DateTx
 P DateText        E
** Monatsnamen
January    February    March       April      May        June
July       August      September   October    November   December
```

Abbildung 3.6: Diese Version der Funktion DateText verfügt über ein eigenes Modul (Teil 2 von 2).

Der wichtigste Unterschied hier ist es, dass die H-Spezifikation für das NOMAIN-Modul das Schlüsselwort NOMAIN enthält. So erfährt der Compiler, dass das Modul keine Main Procedure enthält. Ein weiterer Unterschied ist die Art und Weise, in der Programm und Modul erstellt werden. Da es sich hier um zwei separate Quellen handelt, müssen wir jede Quelle separat kompilieren, wobei wir den Befehl Create RPG Module (CRTRPGMOD) für jede Source wie folgt verwenden:

```
CRTRPGMOD MODULE(Library/DSPDAT) SRCFILE(Library/QRPGLESRC)

CRTRPGMOD MODULE(Library/DATEFN) SRCFILE(Library/QRPGLESRC)
```

Anschließend müssen wir diese beiden Komponenten mit dem Befehl Create Program (CRTPGM) kombinieren.

Die Namen der zu integrierenden Module werden mit Hilfe des Parameters MODULE festgelegt. In unserem Beispiel wird dieser Befehl wie folgt verwendet:

```
CRTPGM PGM(Library/DDT001RG) MODULE(DSPDAT DATEFN) ENTMOD(*FIRST)
```

In diesem Beispiel wird der Parameter Entry Module (ENTMOD) verwendet, um festzulegen, welche der Module, die eine Main Procedure enthalten, als Eingabemodul verwendet werden sollen oder um das erste Modul festzulegen, das beim Aufruf des Programms ausgeführt wird. In den meisten Fällen verfügen Sie nur über eine Prozedur mit einer Main Procedure sowie über null oder mehr Module ohne Main Procedure und nur mit Subprozeduren. Wenn Ihr Programm mehrere Module in einer Main Procedure enthält, müssen Sie diesen Modulnamen im Parameter ENTMOD angeben. Sie können auch optional *ONLY für das Eintragsmodul angeben, um eine Prüfung ausführen zu lassen, mit der sichergestellt wird, dass nur ein Modul im Parameter MODULE ein Eintragsmodul enthält.

Es wird Ihnen sicher auffallen, dass das Modul NOMAIN in Abbildung 3.6 einige Datendefinitionsspezifikationen enthält, die vor der Subprozedur selbst angegeben werden. Sie können globale Variablen erstellen, die allen Subprozeduren innerhalb eines Moduls zur Verfügung stehen. Diese Variablen müssen vor dem Subprozedur-Code definiert werden. In unserem Beispiel wird das Array Months in diese Position eingefügt, weil sich die Werte für Kompilierzeit-Arrays am Ende der Source befinden. Zusätzlich müssen alle Da-

teien, die von den Subprozeduren innerhalb Ihres NOMAIN-Moduls benötigt werden, vor den Subprozedurdefinitionen definiert werden.

Der primäre Zweck der Erstellung eines NOMAIN-Moduls ist es, dass Sie beabsichtigen, die Subprozeduren innerhalb dieses Moduls mehreren Programmen zur Verfügung zu stellen. Daher fragen Sie sich vielleicht: „Was geschieht, wenn ich das Modul ändern muss, das die Subprozedur enthält?“ Die Antwort auf diese Frage ist der Befehl Update Program (UPDPGM). Dieser Befehl aktualisiert ein Modul oder Module innerhalb eines Programms, so dass wir das Programm nicht neu erstellen müssen – wir erstellen einfach das Modul NOMAIN mit Hilfe des Befehls CRTRPGMOD neu. Danach aktualisieren wir alle Programme, die an das Modul gebunden sind, indem wir den Befehl UPDPGM für die einzelnen Programme einsetzen. In unserem Beispiel wird dieser Befehl wie folgt verwendet:

```
UPDPGM PGM(Library/DDT001RG) MODULE(Library/DATEFN)
```

Wenn eine Funktion oder Subprozedur innerhalb eines Moduls die Anzahl der verwendeten Parameter ändert, muss das Programm neu erstellt werden, weil der Prototyp für die Funktion oder die Subprozedur innerhalb des Programms aktualisiert werden muss. Wenn die Änderung jedoch nicht von einer bestimmten Anwendung benötigt wird, können Sie dieses Programm ausschließen. In diesem Fall wird das Programm weiterhin die Version des Moduls verwenden, das bei Erstellung des Programms verfügbar war. Das ist vorteilhaft, weil es bedeutet, dass Sie die Funktionen innerhalb eines Moduls aktualisieren können und nur die Programme aktualisieren, die die neue Funktionalität benötigen. Verges-

sen Sie nicht, dass Sie die Änderungen am Prototyp implementieren müssen, wenn Sie ein bestimmtes Programm nicht aktualisieren und dieses Programm an einem bestimmten Punkt neu erstellt wird. Das ist der einzige Grund, warum es wirklich eine gute Idee ist, die Prototyp-Source in einer eigenen Quellenteildatei zu speichern und die Compiler-Direktive /COPY zu verwenden, um das Modul und erforderliche Programme mit der Source zu ergänzen. Abbildung 3.7 zeigt die Prototyp-Source DATETEXTPR für die Funktion DateText.

```
*----------------------------------------------------------------------*
* DATETEXTPR: DateText Funktionsprototyp
*----------------------------------------------------------------------*
D DateText       PR             35   0
D  DateValue                    12A   VALUE
D  DateFmt                       4A   VALUE OPTION(*NOPASS)
```

Abbildung 3.7: Dieser Prototyp wird in unser Programm mit /COPY integriert.

Der Code ist identisch mit der Prototypversion in den beiden letzten Beispielen. Um ihn in diese Programme einzufügen, ergänzen wir die Prototypzeilen innerhalb des Programms mit der folgenden Zeile:

```
D/COPY QRPGLESRC,DATETEXTPR
```

Diese Zeile sorgt dafür, dass der RPG-Compiler die Source, die in der angegebenen Quellenteildatei spezifiziert wird, in die Source für die Kompilierung noch vor der Kompilierung einfügt. So wird sichergestellt, dass Änderungen, die am Prototyp vorgenommen werden, in sämtlichen Anwendungen, die den Prototyp enthalten,

implementiert werden. Dieser Prozess ist wichtig, um benutzerdefinierte Funktionen in einem ILE RPG-Programm erstellen und verwenden zu können. Nachdem Sie nun eine grundsätzliche Vorstellung haben, sehen wir uns einige praktischen Beispiele für benutzerdefinierte Funktionen an.

Nützliche Programmierbeispiele

Sie werden erkennen, wie sinnvoll es ist, benutzerdefinierte Funktionen zu erstellen, nachdem Sie sich mit den vier separaten Funktionen befasst haben, die Ihnen in diesem Abschnitt vorgestellt werden. Sie werden erfahren, wie diese in ein einziges Modul eingefügt und aus den Anwendungen heraus verwendet werden. Diese Beispiele verwenden einige der erweiterten Anzeigefunktionen, mit denen mehrere nützliche benutzerdefinierte Funktionen erstellt werden.

Dazu gehören:

- Ein Popup-Kalender, der Schaltflächen-Controls verwendet, um einem Benutzer die Auswahl eines Datums zu ermöglichen, das anschließend an die Anwendung zurückgegeben wird.
- Eine Messagebox-Funktion, die weitgehend wie eine Messagebox funktioniert, die in Microsoft Windows verwendet wird.
- Ein Popup-Rechner, der ebenfalls das Schaltflächen-Control verwendet und einem Benutzer die Verwendung von Standardrechnerfunktionen ermöglicht, wobei die Ergebnisse an die aufrufende Anwendung zurückgegeben werden.
- Ein Listenfeld-Control, das mit Hilfe einer Subfile mit Fenstern ein einfaches Listenfeld anzeigt. Diese Funktion ermöglicht es dem Benutzer, ein Element aus der angezeigten Liste zu wählen. Der ausgewählte Wert wird anschließend an die Anwendung zurückgegeben.

Bevor wir beginnen, sehen wir uns die einzelnen Funktionen separat an und prüfen, wie sie funktionieren. Anschließend kombinieren wir die vier Funktionen in ein einzelnes NOMAIN-Modul. Zuletzt erstellen wir eine Beispielanwendung, von der diese Funktionen verwendet werden. Bevor wir uns mit den Funktionen selbst beschäftigen, sehen wir uns die DDS-Source für die Display File an, die von allen vier Funktionen verwendet werden wird. Abbildung 3.8 enthält Bereiche der Display File Source für den Datensatz CALENDARFT in Teildatei WTF001DS. Die gesamte Source kann von der Webseite zu diesem Buch heruntergeladen werden.

```
     *
     *
     * Befehl erstellen: CRTDSPF FILE(LIBRARY/WTF001DS)
     *                           SRCFILE(LIBRARY/QDDSSRC)
     *
    A                                      DSPSIZ(24 80 *DS3)
    A          R CALENDARFT
    A                                      CF12
    A                                      CF03
    A                                      ROLLUP
    A                                      ROLLDOWN
    A                                      WINDOW(10 10 10 47)
    A                                      WDWTITLE((*TEXT &CTITLE))
    A                                  2   1'Sun    Mon    Tue    Wed   Thr  -
    A                                        Fri    Sat   '
    A                                      DSPATR(HI)
    A                                      DSPATR(UL)
    A            DYSLT             2Y 0B  3   1PSHBTNFLD(*RSTCSR (*NUMCOL 7) -
    A                                      (*GUTTER 1))
    A                                      CHCAVAIL((*COLOR RED))
    A                                      PSHBTNCHC(1 &D1)
    A                                      CHCCTL(1 &O1)
    A                                      PSHBTNCHC(2 &D2)
    A                                      CHCCTL(2 &O2)
    A                                      PSHBTNCHC(3 &D3)
    A                                      CHCCTL(3 &O3)
    A                                      PSHBTNCHC(4 &D4)
    A                                      CHCCTL(4 &O4)
```

Abbildung 3.8: Dieser Code konfiguriert das Display-File-Fenster für den Popup-Kalender (Teil 1 von 2).

```
.
.
.
 A                                      CHCCTL(42 &O42)
 A            O1              1Y 0H
 A            O2              1Y 0H
 A            O3              1Y 0H
 A            O4              1Y 0H
.
.
.
 A            O39             1Y 0H
 A            O40             1Y 0H
 A            O41             1Y 0H
 A            O42             1Y 0H
 A                                9  1'ENTER to select        ROLL -
 A                                    Next/Previous Month'
 A                                    DSPATR(UL)
 A                                    DSPATR(HI)
 A            D1               2A  P
 A            D2               2A  P
 A            D3               2A  P
 A            D4               2A  P
.
.
.
 A            D39              2A  P
 A            D40              2A  P
 A            D41              2A  P
 A            D42              2A  P
 A            CTITLE          27A  P
 A            TITLE2          27A  O
```

Abbildung 3.8: Dieser Code konfiguriert das Display-File-Fenster für den Popup-Kalender (Teil 2 von 2).

Dieser Code definiert das Kalenderfenster selbst sowie die Überschriften für die Wochentage. Das Schlüsselwort WDWTITLE wird gemeinsam mit einem Program-to-System-Feld verwendet, um die Übergabe des Titels für den Popup-Kalender an den Bildschirm durch unsere Anwendung zu ermöglichen. Das Schlüsselwort PSHBTNFLD wird verwendet, um ein Gitter aus sechs Zei-

len und sieben Spalten zu erstellen, das aus Schaltflächen-Controls besteht, die für die Anzeige der Tage des Kalendermonats verwendet werden. Die verborgenen Felder *(O1* bis *O42),* die im Schlüsselwort CHCCTL verwendet werden, erlauben uns die Kontrolle, ob eine bestimmte Schaltfläche von innerhalb der Anwendung aus verfügbar ist oder nicht. Die Werte, die in den Schaltflächen selbst angezeigt werden, in diesem Fall die Wochentage, werden mit dem Schlüsselwort PSHBTNCHC sowie mit den verborgenen Feldern *D1* bis*D42* bereitgestellt. Das Feld TITLE2 wird verwendet, um den Monat und das Jahr ganz oben im Kalenderfenster anzuzeigen.

Zusätzlich ist das Fenster MSGBOXFT in dieser Display File enthalten. Dieses Display-File-Fenster wird für das Messagebox-Control verwendet. Abbildung 3.9 zeigt die Source für diesen Display-File-Datensatz.

```
A          R MSGBOXFT
A                                      WINDOW(&Y &X 8 40)
A                                      WDWTITLE((*TEXT &TITLE))
A            RTVAL          2Y 0B  6  3PSHBTNFLD(*RSTCSR (*NUMCOL 3) -
A                                      (*GUTTER 4))
A                                      CHCAVAIL((*COLOR WHT))
A                                      CHCUNAVAIL((*DSPATR ND))
A                                      PSHBTNCHC(1 &BTN1)
A                                      CHCCTL(1 &O1)
A                                      PSHBTNCHC(2 &BTN2)
A                                      CHCCTL(2 &O2)
A                                      PSHBTNCHC(3 &BTN3)
A                                      CHCCTL(3 &O3)
A            TITLE      20A     P
A            X           2S    0P
A            y           2S    0P
```

Abbildung 3.9: Dieser Display-File-Datensatz wird verwendet, um eine Messagebox zu erstellen (Teil 1 von 2).

```
A            BTN1        8A      P
A            BTN2        8A      P
A            BTN3        8A      P
A            TEXT1      40A      O  2  1
A            TEXT2      40A      O  3  1
A            TEXT3      40A      O  4  1
A            01          1Y     0H
A            02          1Y     0H
A            03          1Y     0H
```

Abbildung 3.9: Dieser Display-File-Datensatz wird verwendet, um eine Messagebox zu erstellen (Teil 2 von 2).

Diese Display File verwendet auch ein Schaltflächen-Control, das diesmal nur drei Schaltflächen enthält, die verwendet werden, um die optionalen Antworten auf die angezeigte Meldung zu definieren (Abbrechen, Neuversuch, Ignorieren etc.) Zusätzlich zu den Schaltflächen enthält diese Anzeige auch drei Zeilen mit 40 Zeichen, die für die Anzeige des Meldungstexts verwendet werden. Die Program-to-System-Felder *x* und *y* werden für das Schlüsselwort WINDOW verwendet, um die Einstellung der Position des Fensters im Bildschirm von der aufrufenden Anwendung aus zu ermöglichen.

Abbildung 3.10 zeigt die Source für den Datensatz CALCFT. Dieser Display-File-Datensatz wird von der Popup-Kalenderfunktion verwendet.

```
A          R CALCFT
A                                     WINDOW(&Y &X 10 24)
A                                     WDWTITLE((*TEXT &TITLE))
A                                     CF12
A                                     CF03
A                                     ROLLUP
A                                     ROLLDOWN
A            BUTTN          2Y 0B  4  2PSHBTNFLD(*RSTCSR (*NUMCOL 4))
A                                     CHCAVAIL((*COLOR RED))
A                                     PSHBTNCHC(1 '7')
A                                     PSHBTNCHC(2 '8')
A                                     PSHBTNCHC(3 '9')
A                                     PSHBTNCHC(4 '+')
A                                     PSHBTNCHC(5 '4')
A                                     PSHBTNCHC(6 '5')
A                                     PSHBTNCHC(7 '6')
A                                     PSHBTNCHC(8 '-')
A                                     PSHBTNCHC(9 '1')
A                                     PSHBTNCHC(10 '2')
A                                     PSHBTNCHC(11 '3')
A                                     PSHBTNCHC(12 'x')
A                                     PSHBTNCHC(13 '0')
A                                     PSHBTNCHC(14 '.')
A                                     PSHBTNCHC(15 '=')
A                                     PSHBTNCHC(16 '/')
A                                     PSHBTNCHC(17 'C')
A            DISPLAY       24A  O  2  1
A            ENTRY          1A  B  3  1DSPATR(ND)
A                                     CHECK(ER)
A            TITLE         20A  P
A            X              2S 0P
A            Y              2S 0P
```

Abbildung 3.10: Die Pop-up-Rechnerfunktion wird mit Hilfe dieses Display-File-Datensatzes erstellt.

Diesmal verwenden wir das Schaltflächen-Control als Schaltfläche für unseren Popup-Rechner. Da die Rechnerschaltflächen unverändert bleiben, können wir die entsprechenden Werte im Schlüsselwort PSHBTNCHC hart kodieren. Für zusätzliche Flexibilität können wir hier auch ein einstelliges Feld verwenden, das es dem

Benutzer erlaubt, einen Wert einzutippen, anstatt auf die Schaltfläche zu klicken. Wie beim Meldungsfeld-Fenster können wir die Position des Popup-Rechners mit Hilfe der Program-to-System-Felder *x* und *y* festlegen.

Zuletzt wird der Subfile-Datensatz LISTSF gemeinsam mit dem Subfile-Control-Datensatzfenster LSTITM angegeben, um ein Listenfeld anzuzeigen. Die Source für beide Datensätze sehen Sie in Abbildung 3.11.

```
A              R LISTSF                    SFL
A                                          SFLNXTCHG
A                CTL             1Y 0H     SFLCHCCTL
A                LSTITM         60   0  1  1
A              R LIST                      SFLCTL(LISTSF)
A  50                                      SFLDSP
A  50                                      SFLDSPCTL
A N50                                      SFLDLT
A N50                                      SFLCLR
A  50                                      SFLEND(*SCRBAR *SCRBAR)
A                                          SFLSIZ(0100)
A                                          SFLPAG(0016)
A                                          WINDOW(3 5 17 65)
A                                          WDWTITLE((*TEXT &TITLE))
A                                          SFLSNGCHC(*AUTOSLT)
A                TITLE          20A  P
```

Abbildung 3.11: Diese beiden Anzeigedatei-Datensätze werden für die Erstellung eines Listenfelds verwendet.

Der Subfile-Datensatz selbst enthält ein Feld mit 60 Zeichen, das verwendet wird, um die Werte in unserem Listenfeld anzuzeigen. Dieser Datensatz enthält auch ein Steuerungsfeld mit dem Schlüsselwort SFLCHCCTL. Dieses Feld wird verwendet, um den Wert des ausgewählten Felds zurückzugeben. Der Subfile-Steuerdatensatz LSTITM enthält das Schlüsselwort SFLSNGCHC, das

verwendet wird, um anzugeben, dass die Subfile selbst als ein Einfachauswahl-Control angezeigt werden soll. Damit ermöglichen Sie es dem Benutzer, ein Element in der Liste auszuwählen, indem er einfach den Mauszeiger an die betreffende Stelle verschiebt und die Eingabetaste drückt. Jetzt kennen Sie die Aufgabe der Display File bei der Erstellung unserer Funktionen. Als Nächstes beschäftigen wir uns mit dem Code, der hinter den einzelnen Funktionen steht. Den Anfang macht der Popup-Kalender.

Popup-Kalender

Die Popup-Kalenderfunktion (bezeichnet als Calendar) ist eine hervorragende Alternative zur Eingabe eines Datums durch den Benutzer. Der Vorteil dieser Funktion ist es, dass Sie sich nicht um die Prüfung der Daten kümmern müssen, weil innerhalb des Kalenders nur gültige Datumsangaben angezeigt werden. Im Bildschirm wird ein Monatskalender angezeigt, in dem jeder Tag des Monats mit einem Schaltflächen-Control angezeigt wird. Ein Tag wird ausgewählt, indem der Benutzer auf die Schaltfläche mit dem Mauszeiger klickt oder indem er das gewünschte Datum mit Hilfe der Pfeiltasten markiert und anschließend die Leertaste oder die Eingabetaste drückt. Die Funktion selbst wird mit Hilfe des Codes EVAL op aufgerufen wie in diesem Beispiel:

```
C Eval Date=Calendar(StrDate, Format, Heading)
```

Der erste Parameter ist erforderlich und wird verwendet, um für den Kalender ein Startdatum anzugeben. Diese Informationen werden benötigt, um es der Kalenderfunktion zu ermöglichen, festzustellen, welcher Monat und welches Jahr beim Aufruf angezeigt werden soll. Der zweite Parameter ist optional und wird angegeben, um das Format des Datums in Parameter 1 zu definieren. Tabelle 3.1 zeigt eine Liste der möglichen Werte für Parameter 2 und eine Beschreibung ihrer Verwendung. Den Schluss bildet der dritte Parameter, der ebenfalls optional ist. Dieser Parameter wird spezifiziert, um den Text festzulegen, der in der Titelleiste des Kalenderfensters angezeigt werden soll.

Tabelle 3.1: Gültige Optionen für das Datumsformat in der Kalenderfunktion	
Datumsformat	**Beschreibung**
*MDY	Datum im 6-stelligen Format (Monat, Tag, Jahr)
*YMD	Datum im 6-stelligen Format (Jahr, Monat, Tag)
*CYMD	Datum im 7-stelligen Format (Jahrhundert, Jahr, Monat, Tag)
*MDYY	Datum im 7-stelligen Jahres-, Monats- und Tagesformat
*YYMD	Zeigt die Abfragedefinition mit dem Modus an, in dem die Abfrage zuletzt gespeichert wurde.

Der nächste Parameter, der optional ist, wird angegeben, um einen Titel für das Kalenderfenster zu definieren. Wenn dieser Parameter weggelassen wird, werden Monat und Jahr im Fenstertitelbereich angezeigt. Anderenfalls wird der angegebene Titel verwendet und der Monat und das Jahr werden in der ersten Zeile innerhalb des Fensters angezeigt. Die beiden letzten Parameter werden angegeben, um jeweils die horizontalen und vertikalen Startpositionen des Kalenderfensters im Bildschirm zu definieren. Wenn diese Parameter fehlen, wird eine Standardposition in der Mitte des Bildschirms verwendet.

Diese Funktion verwendet ein Schaltflächenfeld mit zweiundvierzig Schaltflächenoptionen, die unter Verwendung von Program-to-System-Feldern definiert werden, und zweiundvierzig Optionssteuerungsschlüsselwörter, von denen verborgene Felder verwendet werden. Wir verwenden jeweils zweiundvierzig dieser Elemente,

da wir so sechs Reihen mit jeweils sieben Spalten erstellen können. So können wir sieben Tage auf bis zu sechs Reihen verteilen, mit denen Monate mit jeweils fünf Wochen abgebildet werden.

Diese Funktion befasst sich als Erstes mit dem Datum und dem Datumsformat, das angegeben wird, und prüft, ob diese gültig sind. Wenn nicht, wird der Rückgabewert auf *ERROR gesetzt und die Funktion wird beendet. Wenn das Datum und das Datumsformat gültig sind, konvertiert die Funktion das Textfeld, das im ersten Parameter angegeben wird, zu einem Datumsfeld im *ISO-Format. Jahr, Monat und Tag werden danach mit Hilfe des Opcodes EXTRCT extrahiert. Die Kalendersubprozedur verwendet diese Werte in der Subroutine FillCalendar, um festzulegen, auf welchen Wochentag der erste Tag des anzuzeigenden Monats fällt.

Diese Routine füllt dann das Array D (1-42), das die Tagnummerninformationen für die Schaltflächenoptionsschlüsselwörter enthält. Diese Routine aktualisiert auch das Array O (1-42), das im Schlüsselwort Choice Control verwendet wird, um sicherzustellen, dass nur Schaltflächen mit gültigen Tagen für den angezeigten Monat verfügbar sind. Für die Schaltflächen, die keine gültigen Tage enthalten, wird der CHCCTL-Wert auf 4 gesetzt, womit verhindert wird, dass die Schaltfläche aktiviert wird.

Der Benutzer kann zum nächsten oder zum vorherigen Monat mit Hilfe der Rolltasten wechseln. In diesem Fall wird der Wert im Datumsfeld DateFld, der mit dem ersten Parameter angegeben wird, mit jeweils einem Monat addiert bzw. subtrahiert, indem der Opcode ADDUR oder SUBDUR gemeinsam mit dem *M-Extender angegeben wird. Wenn Sie diesen Extender mit einem dieser Befehle nach dem Wert in Faktor 1 positionieren, wird eine

angegebene Anzahl von Monaten mit einem angegebenen Datum addiert oder von diesem Datum subtrahiert. Wenn Sie diesen Wert ersetzen, erfolgt die Inkrementierung in Jahren durch Angabe von *Y oder in Tagen durch Angabe von *D. Wenn der Benutzer ein Datum auswählt, indem er auf den gewünschten Tag des Monats klickt, erstellt das Programm das Datum mit den Monats-, Tages- und Jahresfeldern und gibt diesen Wert an die aufrufende Anwendung im gleichen Format zurück – wie im zweiten Parameter angegeben.

Abbildung 3.12 zeigt die Definitionsspezifikationen für die Calendar-Funktion. Dazu gehört nicht der Code, der außerhalb unserer Subprozeduren erscheint. Denken Sie daran, dass die vier Funktionen, die wir hier entwickeln, in einem Modul enthalten sein werden. Daher werden alle globalen Definitionen vor diesen Subprozeduren positioniert. Dazu gehört die Display File selbst, weil Subprozeduren keine F-Spezifikationen enthalten.

```
     *
     * Kalenderdatums-Auswahlkontrolle
     *
    P Calendar       B                   EXPORT
    D Calendar       PI             8
    D CalendarDate                 12    VALUE
    D DateFmt                       5    VALUE
    D WinTitle                     20    OPTIONS(*NOPASS)
    D RetDate        S              8  0
    D DateA          S              8
    D Pos            S              4  0
    D Len            S              3  0
    D Year4          S              4  0
    D Year3          S              3  0
    D DateFld        S               D   DATFMT(*ISO)
```

Abbildung 3.12: Dieser Code definiert die Felder, die innerhalb unserer Popup-Kalenderfunktion verwendet werden (Teil 1 von 2).

```
D DateJul         S              5  0
D YTDDays         S              3  0
D DateLen         S              1  0
D MonthDay        S              4  0
D MonthDayA       S              4
D Month2          S              2  0
D Mths            S              2  0
D Day2            S              2  0
D DayWk#          S             12  4
D Day#            S              5  0
D Day             S              3  0
D Gap             S              3  0
D LeapCheck       S              3  3
D LeapChkt        S             10  3
D Year4A          S              5
D Stat            S              5  0
D Stat2           S              5  0
D ErrMsg          C                   CONST('Ungültiges Datumsformat bereit-
                                        gestellt')
D
```

Abbildung 3.12: Dieser Code definiert die Felder, die innerhalb unserer Popup-Kalenderfunktion verwendet werden (Teil 2 von 2).

In die Funktion werden zwei erforderliche Parameter und ein optionaler Parameter übergeben. Der erste Parameter *CalendarDat* wird verwendet, um für den anzuzeigenden Kalender ein Startdatum anzugeben. Der zweite Parameter *DateFmt* definiert das Format beider Datumsangaben, d.h. des übergebenen Datums und des Datums, das von der Funktion an das aufrufende Programm zurückgegeben wird. Als Letztes kann der optionale Parameter *WinTitle* verwendet werden, um für das Kalenderfenster einen angepassten Titel zu vergeben. Die übrigen D-Spezifikationen werden einfach verwendet, um die Felder zu definieren, die innerhalb dieser Funktion verwendet werden. Variablen, die innerhalb einer Subprozedur definiert werden, sind nur für diese Subprozedur verfügbar.

Unsere Calculation-Spezifikationen werden verwendet, um den Kalender zu erstellen und das ausgewählte Datum an die aufrufende Anwendung zurückzugeben. Abbildung 3.13 zeigt den Hauptabschnitt der Funktion *Calendar*.

```
C                     Open      WTF001DS
C                     Eval      DateLen=%Len(%Trim(CalendarDate))
C*
C                     If        (DateLen=6 AND (DateFmt'*MDY' And
C                               DateFmt'*YMD' )) Or (DateLen=7 And
C                               DateFmt'*CYMD') Or (DateLen=8 And
C                               (DateFmt'*YYMD' And DateFmt'*MDYY'))
C                               Or (DateLen<6 Or DateLen>8)
C                     Eval      CalendarDate='*ERROR'
C                     Return    CalendarDate
C                     Endif
C*
C                     If        %Parms<3
C                     Eval      WinTitle=*Blanks
C                     EndIf
C                     Exsr      DateFormat
C                     Exsr      FillCalendar
C*
C       Redsp         Tag
C*
C*
C                     Exfmt     CalendarFt
C*
C                     Eval      Stat=%STATUS(WTF001DS)
C                     Eval      Stat2=%STATUS
C*
C                     If        Stat=01122
C                     Exsr      NextMth
C                     Goto      Redsp
C                     Endif
C*
C                     If        Stat=01123
C                     Exsr      PrevMth
C                     Goto      Redsp
C                     Endif
C*
C                     If        *INKC Or *INKL
```

Abbildung 3.13: Dieser Code steuert die Anzeige unseres Popup-Kalenders (Teil 1 von 2).

```
C                   Eval    CalendarDate='00000000'
C                   Else
C                   Eval    Day2=(DYSLT-Day#+1)
C                   Eval    DateA=%Trim(%Char(Year4))+%Trim(%EditC(
C                           Month2:'X'))+%Trim(%EditC(Day2:'X'))
C     *ISO0         Move    DateA         DateFld
C                   Select
C                   When    DateFmt='*YMD'
C     *YMD          Move    DateFld       RetDate
C                   When    DateFmt='*MDY'
C     *MDY          Move    DateFld       RetDate
C                   When    DateFmt='*CYMD'
C     *CYMD         Move    DateFld       RetDate
C                   When    DateFmt='*YYMD'
C     *ISO          Move    DateFld       RetDate
C                   When    DateFmt='*MDYY'
C     *USA          Move    DateFld       RetDate
C                   EndSl
C                   Eval    CalendarDate=%Trim(%Char(RetDate))
C                   EndIf
C                   Return  CalendarDate
```

Abbildung 3.13: Dieser Code steuert die Anzeige unseres Popup-Kalenders (Teil 2 von 2).

Diese Funktion prüft zuerst, welche Tage für die einzelnen Schalter für PSHBTNCTL verwendet werden und zeigt dann das Fenster an. Anschließend kann der Benutzer ein Datum auswählen. Der ausgewählte Wert wird dann in dem Format zurückgegeben, das in dem Parameter festgelegt wird, der in die Funktion übergeben wird. Abbildung 3.14 zeigt die Subroutine FillCalendar, die verwendet wird, um die Schaltflächen-Controls für die Kalenderanzeige zu füllen.

```
C*
C* Subroutine für das Ausfüllen des Kalenders
C*
C      FillCalendar     BegSR
C*
C                       Eval    Day = 0
C*
C                       Eval    LeapChkt=Year4/400
C                       Z-Add   LeapChkt      LeapCheck
C                       If      LeapCheck=0
C                       Eval    MDays(2)=29
C                       Else
C                       Eval    LeapChkt=Year4/100
C                       Z-Add   LeapChkt      LeapCheck
C                       If      LeapCheck=0
C                       Eval    MDays(2)=28
C                       Else
C                       Eval    LeapChkt=Year4/4
C                       Z-Add   LeapChkt      LeapCheck
C                       If      LeapCheck=0
C                       Eval    MDays(2)=29
C                       Else
C                       Eval    MDays(2)=28
C                       EndIf
C                       EndIf
C                       EndIf
C                       Eval    Year3=Year4-1900
C                       If      Day2>MDays(Month2)
C                       Eval    Day2=MDays(Month2)
C                       EndIf
C* Daten-Fehlerbehandlung
C                       If      (Month2 <1 or Month2>12) Or
C                               (Year4<1900 or Year4>2899)
C                       Eval    CalendarDate='*ERROR'
C                       Return  CalendarDate
C                       EndIf
C*
C* Festlegen des Wochentags für den ersten Tag des Monats
C*
C      *JUL             Movel   DateFld       DateJul
C                       Move    DateJul       YTDDays
```

Abbildung 3.14: Dieser Kalender wird für das Füllen der Popup-Kalenderanzeige verwendet (Teil 1 von 2).

```
 C*
 C                     Eval    DayWk#=(Year3*365.25)+(YTDDays)+1
 C                     Eval    Day#=((((DayWk#/7)-%INT(DayWk#/7))*7)+.759)
 C*
 C                     If      Day#=0
 C                     Eval    Day#=7
 C                     EndIf
 C*
 C                     Do      42            X                 2 0
 C                     If      X<Day# OR Day=MDays(Month2)
 C                     Eval    O(X)=4
 C                     Eval    D(X)='  '
 C                     Else
 C                     Eval    Day=Day+1
 C                     If      Day=Day2
 C                     Eval    O(X)=1
 C                     Eval    DYSLT=X
 C                     Else
 C                     Eval    O(X)=0
 C                     EndIf
 C                     Eval    D(X)=%SUBST(%EDITC(Day:'4'):2:2)
 C                     EndIf
 C                     EndDo
 C                     Eval    Year4A = %Char(Year4)
 C                     Clear   Title2
 C                     Clear   CTitle
 C                     If      WinTitle=*Blanks
 C                     Eval    CTitle=%Trim(MNames(Month2))+', '+Year4A
 C                     Eval    Title2=*Blanks
 C                     Else
 C                     Eval    Len=(27-%Len(%Trim(%Trim(MNames(Month2))+',
'
 C                             +Year4A)))/2
 C                     Eval    CTitle=WinTitle
 C                     Eval    Title2=%Subst('                           ':
 C                             1:Len)+%Trim(MNames(Month2))+', '+Year4A
 C
 C                     EndIf
 C*
 C                     EndSR
```

Abbildung 3.14: Dieser Kalender wird für das Füllen der Popup-Kalenderanzeige verwendet (Teil 2 von 2).

Diese Subroutine bestimmt als Erstes, auf welchen Wochentag der erste Tag des benötigten Monats fällt. Als Nächstes verwendet sie diese Information, um die erforderlichen Schaltflächen für die einzelnen Tage des gewünschten Monats zu füllen. Diese Subroutine formatiert außerdem den Monatsnamen und das Jahr für die Anzeige als Überschrift über den Kalender.

Die drei Subroutinen, die in Abbildung 3.15 gezeigt werden, werden verwendet, um zum vorherigen oder nächsten Monat zu wechseln, wenn der Benutzer die Taste „Nach oben“ oder „Nach unten“ drückt, sowie für die Extrahierung des Datums von seinem angegebenen Format in Monats-, Tag- und Jahresfelder.

```
C*
C*
C* Subroutine für den Wechsel zum vorherigen Monat
C*
C     PrevMth       BegSR
C*
C                   SubDur    1:*M          DateFld
C                   Extrct    DateFld:*Y    Year4
C                   Extrct    DateFld:*M    Month2
C                   Extrct    DateFld:*D    Day2
C                   Exsr      FillCalendar
C*
C                   EndSr     C*
C*
C* Subroutine für den Wechsel zum nächsten Monat
C*
C      NextMth      BegSR
C*
C                   AddDur    1:*M          DateFld
C                   Extrct    DateFld:*Y    Year4
C                   Extrct    DateFld:*M    Month2
C                   Extrct    DateFld:*D    Day2
C                   Exsr      FillCalendar
```

Abbildung 3.15: Diese Subroutinen werden verwendet, um Datumsangaben zu formatieren und von einem Monat zum nächsten zu wechseln (Teil 1 von 2).

```
C*
C                   EndSR
C*
C*
C* Subroutine für die Formatierung des eingegebenen Datums
C*
C      DateFormat    BegSR
C*
C                   EvalR     CalendarDate=%Trim(CalendarDate)
C                   Select
C                   When      DateFmt='*YMD'
C      *YMD0        Move      CalendarDate  DateFld
C                   When      DateFmt='*MDY'
C      *MDY0        Move      CalendarDate  DateFld
C                   When      DateFmt='*CYMD'
C      *CYMD0       Move      CalendarDate  DateFld
C                   When      DateFmt='*YYMD'
C      *ISO0        Move      CalendarDate  DateFld
C                   When      DateFmt='*MDYY'
C      *USA0        Move      CalendarDate  DateFld
C                   EndSl
C*
C                   Extrct    DateFld:*Y    Year4
C                   Extrct    DateFld:*M    Month2
C                   Extrct    DateFld:*D    Day2
C*
C                   EndSR
C*
C*
P Kalendar        E
```

Abbildung 3.15: Diese Subroutinen werden verwendet, um Datumsangaben zu formatieren und von einem Monat zum nächsten zu wechseln (Teil 2 von 2).

Diese Funktion kann Ihre sonst etwas langweiligen Green-Screen-Anwendungen wirklich auffrischen. Aber der größte Vorteil ist ihre Unkompliziertheit. Wenn Ihre Anwendung einen Benutzer dazu auffordert, ein Datum anzugeben, können Sie einfach eine Befehlstaste in Ihre Display File (F4=Calendar) einfügen und die Funktion wie folgt aufrufen:

```
C     If      *INKD
C     Eval    StrDate=Calendar(%Char(UDATE):'*MDY':
C             'Startdatum auswählen')
C     EndIf
```

Dieses Beispiel liefert das aktuelle Sitzungsdatum, das in ein Zeichenformat konvertiert wird, das Datumsformat *MDY und Select Start Date für den Fenstertitel. Abbildung 3.16 zeigt ein Beispiel dafür, wie eine Kalenderfunktionsanzeige aussieht.

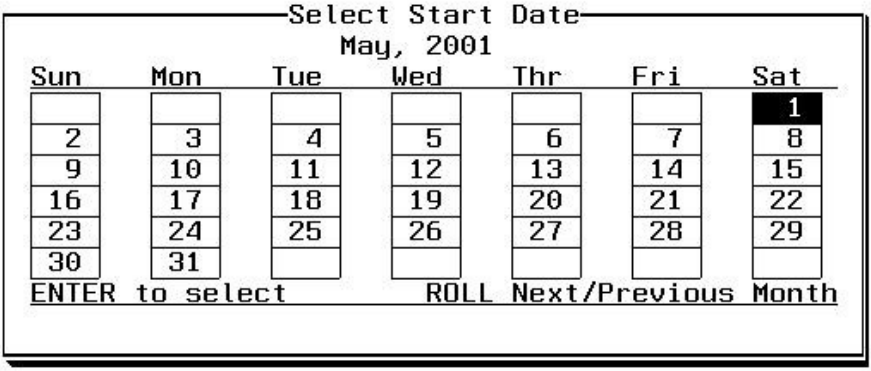


Abbildung 3.16: Die benutzerdefinierte Funktion Calendar erstellt einen Popup-Kalender.

Messagebox

Die Funktion MsgBox ermöglicht es Ihnen, eine Messagebox im Fensterstil fast genauso wie das entsprechende Feld in Visual Basic zu verwenden. Der Code, mit dem der Meldungsfeld-Stil programmiert wird sowie die zurückgegebenen Werte sind genau genommen identisch mit dem Code in VB-Meldungsfeldern. Die Funktion hat fünf Parameter, wobei die ersten drei dieser Parameter erforderlich sind. Der erste Parameter legt den Titel für das Messagebox-Fenster fest. Der zweite Parameter wird verwendet, um den Nachrichtentext selbst zu definieren (mit bis zu 120 Zeichen).

Der dritte Parameter wird angegeben, um den Typ Messagebox für die Anzeige festzulegen. Dieser Parameter bestimmt im Wesentlichen, welche Schaltflächen angezeigt werden. Die beiden letzten Parameter sind optional und werden benötigt, um die horizontalen und vertikalen Positionen des Messagebox-Fensters zu definieren. Tabelle 3.2 enthält eine Liste der Messagebox-Typen und die Rückgabewerte für jede Taste.

Tabelle 3.2: Messagebox-Typen für die Funktion MsgBox		
Typ-Code	**Schaltflächen**	**Rückgabewerte**
1	Ok	1
2	Ok, Abbrechen	1, 2
3	Abbruch, Neuversuch, Ignorieren	3, 4, 5
4	Ja, Nein, Abbrechen	6, 7, 2
5	Ja, Nein	6, 7
6	Neuversuch, Abbrechen	4, 2

Die Werte, die in der Typ-Code-Spalte angezeigt werden, werden als der dritte Parameter in der MsgBox-Funktion verwendet, um festzulegen, welche Schaltflächen angezeigt werden sollen. Die Rückgabewerte zeigen, welche Daten die Funktion an Ihre Anwendung zurückgibt. Diese Funktion ist ein gutes Beispiel dafür, wie benutzerdefinierte Funktionen verwendet werden können, um den Anwendungsfluss zu verbessern.

Es folgt ein Beispiel für die Verwendung dieser Funktion durch Ihre Anwendung:

```
C           If      %Found(CUSTMAST)=*Off
C           Eval    Reply=MsgBox('Datensatz nicht gefunden':
C                   'Möchten Sie einen neuen Datensatz erstellen?':5)
C           Select
C           When    Reply=6
C           Exsr    CreateRecord
C           When    Reply=7
C           Iter
C           EndIf
```

In diesem Beispiel wird die integrierte Funktion %FOUND verwendet, um festzustellen, ob in der letzten Kette ein Datensatz für die Datei CUSTMAST gefunden wurde. Wenn %FOUND deaktiviert ist, wurde kein Datensatz gefunden. In diesem Fall wird das Meldungsfeld angezeigt, das Sie in Abbildung 3.17 sehen, und das den Benutzer darüber informiert und ihn fragt, ob er einen neuen Datensatz erstellen möchte. Wir verwenden Messagebox-Typ 5, der eine Option „Ja“ und eine Option „Nein“ mit einschließt. Wenn der Benutzer „Ja“ wählt, wird als Antwort in die Variable Reply eine 6 zurückgegeben. In diesem Fall rufen wir die Subroutine auf, um einen neuen Datensatz zu erstellen. Wenn der Benutzer „Nein“ wählt, wird die Variable Reply auf 7 gesetzt und wir kehren ganz an den Anfang der aktuellen DO-Schleife zurück.

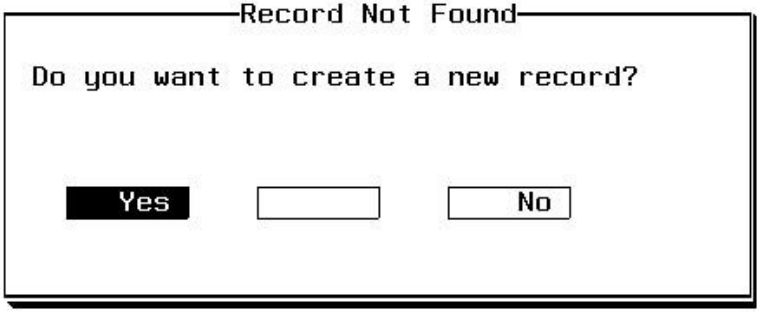


Abbildung 3.17: Die Funktion MsgBox wird verwendet, um Nachrichten anzuzeigen und eine Antwort zu akzeptieren.

Der Code für die Funktion MsgBox wird in Abbildung 3.18 gezeigt. Diese Abbildung zeigt nur den Code für die Subprozedur selbst. Wie Sie sehen, ist die eigentliche Funktion recht einfach, aber die Funktionen, die von ihr bereitgestellt werden, können das Aussehen einer Anwendung komplett verändern.

```
 *
 * Messagebox-Control
 *
P MsgBox          B                   EXPORT
D MsgBox          PI             5I 0
D  TitleIn                       20    VALUE
D  Message                      120    VALUE
D  Type                           1  0 VALUE
D  Xpos                           3  0 VALUE options(*NOPASS)
D  Ypos                           2  0 VALUE options(*NOPASS)
Dret              S               1  0
C                   If        %Parms<5
C                   Eval      X=10
C                   Eval      Y=10
C                   Else
C                   Eval      X=Xpos
C                   Eval      Y=YPos
C                   EndIf
C                   Do        3             Ix                2 0
C                   If        RtnVals(((Type-1)*3)+Ix)=0
C                   Eval      ShowBtn(Ix)=4
C                   Else
C                   Eval      ShowBtn(Ix)=1
C                   EndIf
C                   EndDo
C                   Eval      Title=TitleIn
C                   Eval      Button=Buttons(Type)
C                   Eval      Messag=Message
C                   Open      WTF001DS
C                   Exfmt     MSGBOXFT
C                   Eval      Ret=RtnVals(((Type-1)*3)+Rtval)
C                   Close     WTF001DS
C                   Return    Ret
P MsgBox          E
```

Abbildung 3.18: Dies ist der MsgBox-Subprozedur-Abschnitt der Fenster-Tool-Funktionen.

Zusätzlich zur Veränderung des Aussehens einer Anwendung kann diese Funktion auch den Programmfluss der Anwendung verändern. Bei unserem früheren Beispiel würden wir ohne die Funktion MsgBox eventuell den Bildschirm erneut ausgeben, der mit einer Fehlernachricht angezeigt wurde, anschließend müsste der Benutzer eine separate Taste drücken, um den neuen Datensatz zu erstellen. Das würde zusätzliche Codezeilen bedeuten und die Programmlogik würde weniger flüssig verlaufen. Diese einfache benutzerdefinierte Funktion bietet hier einen geeigneten Ersatz.

Popup-Rechner

Als Nächstes befassen wir uns mit dem Popup-Rechner-Control. Diese Funktion ist äußerst leistungsfähig, da sie Ihnen erlaubt, einem Benutzer einen Rechner mit Schaltflächen-Feldsteuerung zur Verfügung zu stellen. Dieses Fenster sieht wie ein Standardtaschenrechner aus und verhält sich auch so, davon ausgenommen die Tatsache, dass das aktuelle Ergebnis an die aufrufende Anwendung zurückgegeben wird, wenn der Benutzer die Taste F3 drückt. Abbildung 3.19 enthält die D-Spezifikationen für die Subprozedurfunktion Calc.

```
P Calc            B                   EXPORT
D Calc PI                       24  8
D  DecimalPlaces                 3  0 VALUE
D  TitleIn                      20    VALUE options(*NOPASS)
D  Xpos                          3  0 VALUE options(*NOPASS)
D  Ypos                          2  0 VALUE options(*NOPASS)
D  ValidEntry     C                   CONST('789+456-123*0.=/C')
D  Factor         S              15  5
D  Fact1          S              15  5
D  Result         S              24  8
D  Integer        S              16  0
D  Decimal        S               8  8
D  IntA           S              16
D  DecA           S               8
D  DecPnt         S               1
D  LastOp         S               1
D  LstEn          S               1
D  DecSpot        S               2  0
```

Abbildung 3.19: Die Definitionsspezifikationen für den Popup-Rechner.

Der erste Parameter für diese Funktion ist erforderlich und wird verwendet, um die Dezimalstellen festzulegen, die für die Rechner-

anzeige und für den Wert verwendet werden sollen, der von der Funktion zurückgegeben wird. Die übrigen Parameter sind optional und werden verwendet, um den Titel und die Positionierung des Popup-Fensters zu definieren. Abbildung 3.20 enthält den Hauptcode für die Subprozedur CALC.

```
C              If      %Parms<3
C              Eval    X=10
C              Eval    Y=10
C              Else
C              Eval    X=Xpos
C              Eval    Y=Ypos
C              EndIf
C              If      %Parms<2
C              Eval    Title='Calculator'
C              Else
C              Eval    Title=TitleIn
C              EndIf
C              Open    WTF001DS
C              DoW     *INKC = *OFF
C              Clear                 Entry
C              Clear                 Button
C              ExFmt   CalcFt
C*
C              If      *INKC = *ON
C              Move    Result        Decimal
C              Movel   Result        Integer
C              Move    Decimal       DecA
C              Eval    DecA=%Subst(DecA:1:DecimalPlaces)
C              Move    DecA          Decimal
C              Eval    Result=Integer+Decimal
C              Return  Result
C              Open    WTF001DS
C              EndIf
C              If      Entry='.' And DecPnt=*On
C              Eval    Entry=''
C              EndIf
C*
C              If      Entry ' '
C              If      Entry='x' or Entry = 'X'
C              Eval    Entry = '*'
C              EndIf
```

Abbildung 3.20: Die Calc-Funktion wird in Ihrem Programm wie ein Taschenrechner verwendet (Teil 1 von 3).

```
C               Eval    Buttn=%SCAN(Entry: ValidEntry)
C               EndIf
C*
C               If      Button0
C               Eval    Entry = %Subst(ValidEntry : Buttn: 1)
C               If      (Entry>='0' AND Entry<='9') Or Entry='.'
C               If      LstEn='+' Or LstEn='-' Or LstEn='=' Or
C                       LstEn='*' Or LstEn='/'
C               Eval    Display=''
C               EndIf
C               EvalR   Display=Display+Entry
C               If      Entry='.'
C               Eval    DecPnt=*On
C               EndIf
C               Else
C               Eval    DecPnt=*Off
C               Eval    DecSpot=%Scan('.':%Trim(Display):1)-1
C               If      DecSpot<1
C               If      %Subst(%Trim(Display):1:1)='.'
C               Eval    DecSpot=0
C               EvalR   IntA='0'
C               Else
C               Eval    DecSpot=%Len(%Trim(Display))
C               EvalR   IntA=%Subst(%Trim(Display):1:DecSpot)
C               EndIf
C               Else
C               EvalR   IntA=%Subst(%Trim(Display):1:DecSpot)
C               EndIf
C               If      DecSpot<%Len(%Trim(Display))-1
C               Eval    DecA=%Trim(%Subst(%Trim(Display):DecSpot+2:
C                       %Len(%Trim(Display))-(DecSpot+1)))
C               Else
C               Eval    DecA=*Blanks
C               EndIf
C               Move    IntA          integre
C               Move    DecA          Decimal
C               Eval    Factor=Integer+Decimal
C               Select
C               When    LastOp='*'
C               Eval(h) Result=(Fact1*Factor)
C               When    LastOp='+'
C               Eval    Result=(Fact1+Factor)
C               When    LastOp='/'
C               Eval(h) Result=(Fact1/Factor)
```

Abbildung 3.20: Die Calc-Funktion wird in Ihrem Programm wie ein Taschenrechner verwendet (Teil 2 von 3).

```
C               When    LastOp='-'
C               Eval    Result=(Fact1-Factor)
C               EndSl
C               If      Result=0
C               Eval    Result=Factor
C               EndIf
C               Select
C               When    Entry='='
C               EvalR   Display=%Char(Result)
C               Eval    Fact1=Result
C               When    Entry='C'
C               Eval    Result=0
C               Eval    Display=*Blanks
C               Eval    Factor=0
C               Eval    Entry=''
C               EndSl
C               Eval    LastOp=Entry
C               Eval    Fact1=Result
C               EndIf
C               Eval    LstEn=Entry
C               EndIf
C               EndDo
P CALC E
```

Abbildung 3.20: Die Calc-Funktion wird in Ihrem Programm wie ein Taschenrechner verwendet (Teil 3 von 3).

Diese Funktion ermöglicht es dem Benutzer, entweder die Werte über die Tastatur einzugeben oder die Schaltflächen im Rechner selbst zu verwenden. Addition, Subtraktion, Multiplikation und Division können alle erfolgen, wenn diese Funktion angezeigt wird. Die Ergebnisse werden ganz oben im Rechnerfenster angezeigt. Dieses Ergebnis wird außerdem an die aufrufende Variable zurückgegeben, wobei der Befehl EVAL wie folgt verwendet wird:

```
C Eval Amount=Calc(DecPlcs:'Calculator':10:10)
```

Diese einzelne Zeile sorgt dafür, dass das Rechner-Control angezeigt und beim Beenden der berechnete Wert zurückgegeben wird. Der erste Parameter wird verwendet, um die Anzahl der Dezimalstellen zu definieren, die von 0 bis 8 zurückgegeben werden sollten. Dies ist der einzige erforderliche Parameter. Der zweite Parameter wird verwendet, um einen Titel für das Rechnerfenster zu definieren. Die beiden nächsten Parameter definieren die horizontale und vertikale Startposition für das Rechnerfenster. Wie bei den anderen Funktionen werden Sie feststellen, dass die Fähigkeit, eine Funktion wie diese in Ihre Anwendung einzufügen, Ihren Code stark vereinfachen kann. Abbildung 3.21 zeigt das Fenster, das von der Funktion Calc präsentiert wird.

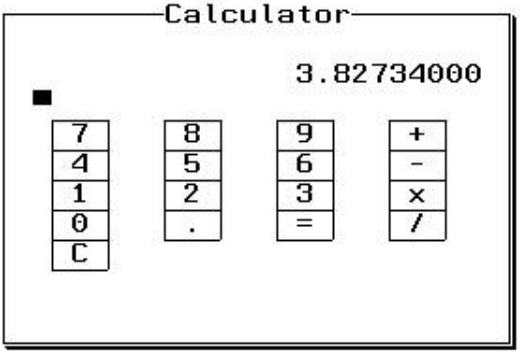


Abbildung 3.21: Anzeige der benutzerdefinierten Funktion Calc.

List Box

Die letzte Funktion, die Ihnen in diesem Kapitel vorgestellt wird, ist die Funktion ListBox. Der Aufruf dieser Funktion von der Anwendung aus ist äußerst kompliziert, weil die erforderlichen Parameter der Funktion die Daten beinhalten, die im Listenfeld in Form eines Arrays angezeigt werden sollen. Diese Funktion zeigt eine Liste der übergebenen Werte in einem Fenster an, das einen Einfachauswahl-Subfile-Datensatz enthält. Der zurückgegebene Wert wird vom zweiten und dritten Parameter bestimmt. Diese definieren die Startposition innerhalb des ausgewählten Datensatzes, der zurückgegeben werden soll, sowie die Länge der zurückzugebenden Daten. Der erste Parameter muss als ein Feld angegeben werden, das bis zu 500 Werte mit einer Maximallänge von 60 enthält. Abbildung 3.22 zeigt die Source für die ListBox-Subprozedurfunktion.

```
P ListBox         B                   EXPORT
D ListBox         PI            60
D  RecordSource                 30000    VALUE
D  WinTitle                        20    VALUE OPTIONS(*NOPASS)
D  StartPos                         2  0 VALUE OPTIONS(*NOPASS)
D  Length                           2  0 VALUE OPTIONS(*NOPASS)
D ListItems       S                 60    DIM(500)
D SelVal          S                 60
 *
C                   Movea     RecordSource  ListItems
C                   Eval      *IN50=*Off
C                   Eval      RRN=0
C                   Eval      Title=WinTitle
C                   Open      WTF001DS
 *
C                   Write     LIST
```

Abbildung 3.22: Die Funktion ListBox wird für die Anzeige einer Liste und die Übernahme einer einzelnen Auswahl verwendet (Teil 1 von 2).

```
 *
C                   Do         500           X                3 0
C                   If         ListItems(X)''
C                   Eval       LSTITM=ListItems(X)
C                   Eval       RRN=RRN+1
C                   Write      LISTSF
C                   EndIF
C                   EndDo
 *
C                   If  RRN=0
C                   Eval       SelVal='*ERROR'
C                   Else
C                   Eval       *IN50=*On
C                   Exfmt      LIST
C                   DoW        %EOF(WTF001DS)=*Off
C                   ReadC      LISTSF
C                   If         Not %EOF(WTF001DS) And CTL=1
C                   Eval       SelVal=%Subst(LSTITM:StartPos:Length)
C                   EndIf
C                   EndDO
C                   EndIf
C                   Close      WTF001DS
 *
C                   Return     SelVal
P ListBox         E
```

Abbildung 3.22: Die Funktion ListBox wird für die Anzeige einer Liste und die Übernahme einer einzelnen Auswahl verwendet (Teil 2 von 2).

Wie bei der Funktion MsgBox ist die Subprozedur selbst nicht sehr kompliziert. Wichtig ist, dass Sie Ihnen eine einfache Methode in die Hand gibt, mit der allgemein verwendete Funktionalität mehreren Programmen zur Verfügung gestellt werden kann. Diese Funktion funktioniert ähnlich wie die in Kapitel 2 erstellte Liste, mit der die Liste der US-Staaten angezeigt wurde.

Abbildung 3.23 zeigt ein Beispiel für die Ausgabe, die bei Verwendung dieser Funktion angezeigt wird.

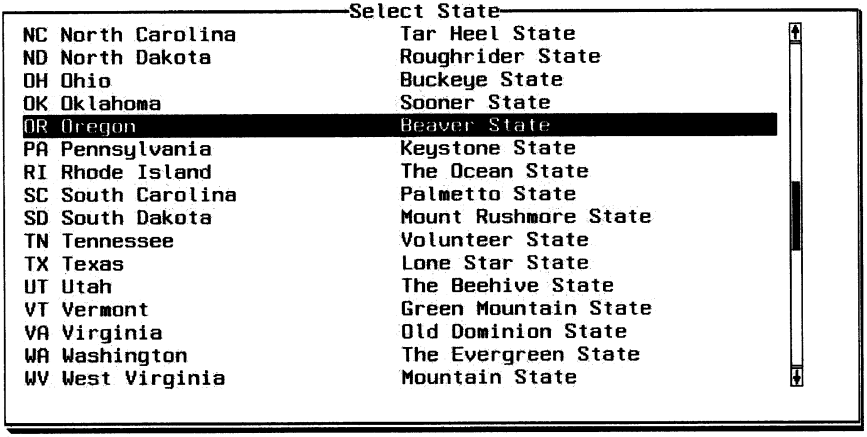


Abbildung 3.23: Dieser Bildschirm wird von der Funktion ListBox erstellt.

Um dieses Fenster zu erstellen, müssen wir zuerst ein Array mit den Werten füllen, die angezeigt werden sollen. Dieses Array wird anschließend in ein Feld mit einer Länge verschoben, die der Gesamtlänge aller Elemente des Arrays entspricht. Wenn wir z.B. 15 Datensätze anzeigen möchten, die je eine Länge von 50 aufweisen, erstellen wir ein Feld mit der Länge von 750. Das Array wird dann in dieses Feld wie folgt verschoben:

```
C MoveA ListArr ListStr
```

Eine einfache Methode für die Konkatenierung der Werte für die Verwendung für die einzelnen Datensätze in der Liste besteht darin, eine Datenstruktur zu erstellen, die Felder mit der richtigen Länge enthält.

Dies geschieht wie folgt:

```
     DlistArElm
     D  State          1     2
     D  StatName       3    30
     D  MickName      35    30
```

Diese Datenstruktur wird anschließend in das aktuelle Element des Arrays verschoben, das in die ListBox gesendet werden soll. Wie bereits erwähnt, erfordert diese Funktion etwas mehr Vorbereitungsarbeiten als die anderen drei vorgestellten Funktionen. Davon abgesehen sind die Ergebnisse ausgezeichnet. Die Startposition und die Längenparameter der Funktion ListBox werden verwendet, um den Bereich des ausgewählten Datensatzes festzustellen, der an die aufrufende Anwendung zurückgegeben werden soll. Wenn für diese kein Wert angegeben wird, wird der gesamte Datensatz zurückgegeben. Wir verwenden ein Programm wie folgt, um die in Abbildung 3.23 gezeigten Ergebnisse zu erzielen.

```
      /COPY QRPGLESRC,WTF001PR
     Dlist             S            60A   DIM(50) CTDATA PERRCD(1)
     DSPOS             S             2  0
     DLNGTH            S             2  0
     Dstate            S             2
     DlistX            S          3000
     C                   Eval     SPOS=1
     C                   Eval     LNGTH=2
     C                   MoveA    List          ListX
     C                   Eval     State=ListBox(ListX:'Select State':
     C                            SPOS:LNGTH)
     C    State          Dsply
     C                   Return
** State names table
AL Alabama                    Yellowhammer State
AK Alaska                     The Last Frontier
AZ Arizona                    The Grand Canyon State
```

Dieses Array wird aus einer Kompilierzeit-Tabelle geladen, aber sie kann auch genauso gut mit Daten aus einer physischen Datei geladen werden. Die Zeile /COPY am Anfang der Funktion wird verwendet, um den Prototyp für die Subprozedurfunktionen ins Spiel zu bringen. Abbildung 3.24 enthält den Prototyp für die einzelnen hier besprochenen Funktionen.

```
 * Teildateinamen: WTF001PT
 *
 * Beschreibung: Copybook-Teildatei für Fenstertool-Funktion
 *
D MsgBox           PR              5I 0
D  Title                          20    VALUE
D  Message                       120    VALUE
D  Type                            1  0 VALUE
D  XPos                            3  0 VALUE OPTIONS(*NOPASS)
D  YPos                            2  0 VALUE OPTIONS(*NOPASS)
* Kalenderdatum-Auswahlprototyp
D Calendar         PR              8
D  CalendarDate                   12    VALUE
D  DateFmt                         5    VALUE
D  WinTitle                       20    OPTIONS(*NOPASS)
D  XP1                             3  0 OPTIONS(*NOPASS)
D  YP1                             2  0 OPTIONS(*NOPASS)
D Calc PR                         24  8
D  DecimalPlaces                   3  0 VALUE
D  TitleIn                        20    VALUE options(*NOPASS)
D  Xpos                            3  0 VALUE options(*NOPASS)
D  Ypos                            2  0 VALUE options(*NOPASS)
D ListBox      PR                 60
D  RecordSource                30000    VALUE
D  WinTitle                       20    VALUE OPTIONS(*NOPASS)
D  StartPos                        2  0 VALUE OPTIONS(*NOPASS)
D  Length                          2  0 VALUE OPTIONS(*NOPASS)
```

Abbildung 3.24: Dies ist der Prototyp für das Fenstertool-Funktionsmodul.

Das Schlüsselwort VALUE ermöglicht uns die Übergabe von Parametern nach Wert. Das ermöglicht es uns, Konstanten anstelle von Feldern zu verwenden, um Parameterwerte für eine Funktion festzulegen. Wie bereits erwähnt, wird mit dem Schlüsselwort OPTIONS(*NOPASS) angegeben, ob ein Parameter optional ist.

Zum Schluss beschäftigen wir uns mit den Bereichen des Moduls WTF001RG, den wir bisher noch nicht kennen. Dazu gehört alles, was sich außerhalb der Subprozedurfunktionen selbst befindet. Alle globalen Werte sowie die Definition für die Display File befinden sich in der Kopfzeile unseres Moduls. Die Daten für die Kompilierzeit-Arrays befinden sich am Ende unserer Modul-Source. Abbildung 3.25 zeigt Bereiche dieses Teils des Moduls.

```
H NOMAIN
 * Programnnamen: WTF001RG
 *
 * Beschreibung:  Window Tool Functions Module
 *
 * Befehle erstellen: CRTRPGMOD MODULE(library/WTF001 )
 *                              SRCFILE(library/QRPGLESRC)
 *
FWTF001DS  CF   E               WORKSTN USROPN
F                                       SFILE(LISTSF:RRN)
D Buttons         S               30    DIM(6) CTDATA PERRCD(1) Btn Captions
D ShowBtn         S                1  0 DIM(3)                 Btn Dsply Ctl
D RtnVals         S                1  0 DIM(18) CTDATA PERRCD(18) Return Values
D MNames          S               10    DIM(12) CTDATA PERRCD(6) Month Names
D MDays           S                2  0 DIM(12) CTDATA PERRCD(12) Days Per Month
D O               S                1  0 DIM(42)
D RRN             S                5  0
D                 DS
DD1                         1       2
DD2                         3       4
DD3                         5       6
DD4                         7       8
.
```

Abbildung 3.25: Die Kopf- und Fußzeilenbereiche des WTF001RG Moduls (Teil 1 von 3).

```
.
.
 DD38                      75      76
 DD39                      77      78
 DD40                      79      80
 DD41                      81      82
 DD42                      83      84
 D D                        1      84   DIM(42)
 D Button         DS                 24
 D Btn1                     1       8
 D Btn2                     9      16
 D Btn3                    17      24
 DMessag          DS
 D   Text1                  1      40
 D   Text2                 41      80
 D   Text3                 81     120
 D/COPY QRPGLESRC,WTF001PR
 IMsgBoxFt
 I              01                          ShowBtn(1)
 I              02                          ShowBtn(2)
 I              03                          ShowBtn(3)
 ICALENDARFT
 I              01                          O(1)
 I              02                          O(2)
 I              03                          O(3)
 I              04                          O(4)
.
.
.
 I              038                         O(38)
 I              039                         O(39)
 I              040                         O(40)
 I              041                         O(41)
 I              042                         O(42)
  *
  * Hier wird die Source für die Subprozedurfunktionen eingefügt.
  *
 ** Schaltflächentext für MSGBOX
         OK
   OK           Cancel
  Abort  Retry  Ignore
   Yes    No    Cancel
   Yes            No
  Retry         Cancel
```

Abbildung 3.25: Die Kopf- und Fußzeilenbereiche des WTF001RG Moduls (Teil 2 von 3).

```
** Schaltflächentext für MSGBOX-Schaltflächen
010102345672607402
** Monatsnamen
January        February March    April    May      June
July   August   September October        November December
** Tage pro Monat
312831303130313130313031
```

Abbildung 3.25: Die Kopf- und Fußzeilenbereiche des WTF001RG Moduls (Teil 3 von 3).

Um das ILE RPG-Funktionsmodul für das Fenster zu erstellen (WTF001RG), verwenden Sie den Befehl RPG Module (CRTRPGMOD) wie hier:

```
CRTRPGMOD MODULE(WTF001RG) SRCFILE(Library/QRPGLESRC)
```

Um einige dieser Module von innerhalb einer Anwendung aus zu verwenden, erstellen Sie als Erstes das Modul für diese Anwendung, indem Sie den Befehl CRTRPGMOD verwenden. Als Nächstes bindet der Befehl Create Program (CRTPGM) die beiden Module. Dafür sorgt dieser Code:

```
CRTPGM PGM(pgmname) MODULE(pgmmodule WTF001RG) ENTMOD(*FIRST)
```

Nachdem der Befehl ausgeführt wurde, verfügen Sie über ein ausführbares Programm, das ein Window-Tools-Funktionsmodul enthält.

4

Advanced Function Printing (AFP)

Die erweiterten Anzeigefunktionen für Ihre Display-Ausgabe finden ihre Entsprechung im Advanced Function Printing (AFP) für Ihre Druckausgaben. AFP ermöglicht es Ihnen, Ihre Druckerdateien auf der 400 mit Zeilen, Feldern, Barcodes und sogar Grafiken zu erweitern – vorausgesetzt, Sie senden die Ausgabe zu einem Drukker, der entweder AFP-kompatibel oder der mit der Hewlett-Packard Printer Control Language (HP-PCL) kompatibel ist. Drukker vom Typ Intelligent Printer Data Stream (IPDS) sind z.B. AFP-kompatibel. Diese Sprache wird auf den meisten Druckern von IBM verwendet. Die meisten Laserdrucker hingegen sprechen HP-PCL. Diese Sprache ist neben Postscript der Standard für Laserdrucker.

OS/400 verfügt über die entsprechende Funktionalität, um eine Druckerdatei unter Verwendung von Host Print Transform (HPT) von AFP in HP-PCL zu übersetzen. Sie können Ihre Druckdateien mit diesen Funktionen ergänzen, indem Sie einfach einige DDS-Schlüsselwörter angeben. Wir befassen uns in diesem Kapitel mit der Frage, welche Konfiguration für die Verwendung von AFP benötigt wird – sowie mit den Schlüsselwörtern und ihrer Verwendung. Außerdem erstellen wir einige Beispielausgaben, mit denen Sie Ihre Freunde und Kollegen beeindrucken können.

Konfiguration

Die Konfiguration für die Verwendung von AFP erfordert nur wenige Schritte. Wenn Sie einen Drucker benutzen, der mit AFP kompatibel ist, müssen Sie nur sicherstellen, dass in der Gerätebeschreibung für den Drucker der AFP-Parameter *YES eingestellt ist. Sie können das prüfen, indem Sie den Befehl Display Device Description (DSPDEVD) wie folgt eingeben:

```
DSPDEVD DEV(Druckernamen)
```

Suchen Sie jetzt nach dem Parameter mit der Bezeichnung Advanced Function Printing. Wenn dieser Parameter auf *YES gesetzt wird, sind Sie soweit. Wenn nicht, müssen Sie die Gerätebeschreibung ändern. Um den Writer zu beenden, verwenden Sie den Befehl ENDWTR wie hier:

```
ENDWTR WTR(Druckernamen) OPTION(*IMMED)
```

Nun beenden Sie die Gerätebeschreibung mit Hilfe des Befehls VRYCFG wie folgt:

```
VRYCFG CFGOBJ(Druckernamen) CFGTYP(*DEV) STATUS(*OFF)
```

Als Nächstes verwenden Sie den Befehl Change Device Description (CHGDEVD), um den Advanced-Function-Printing-Parameter Use wie hier auf *YES zu ändern:

```
CHGDEVD DEV(Druckernamen) AFP(*YES)
```

Aktivieren Sie das Gerät und starten Sie den Writer erneut, indem Sie diese beiden Befehle eingeben:

```
VRYCFG CFGOBJ(Druckernamen) CFGTYP(*DEV) STATUS(*ON)
STRPRTWTR WTR(Druckernamen)
```

Ihr AFP-Drucker sollte nun startbereit sein. Wenn Sie einen Drukker verwenden, der AFP nicht unterstützt, aber HP-PCL-kompatibel ist, weicht die Vorgehensweise vom beschriebenen Prozess leicht ab. Erstens ist auf Nicht-AFP-fähigen Druckern der Use-Advanced-Function-Printing-Parameter nicht verfügbar. Das ist aber kein Problem, weil Sie die Ausgabe vor dem Senden zum Drucker in die HP-PCL-Sprache konvertieren, wenn Sie den Parameter Use Host Print Transform auf *YES setzen und den richtigen Wert für den Parameter „Manufacturer Type and Model“ (MFTPMDL) angeben. Bei einem HP LaserJet 5-Drucker geben Sie dazu Folgendes ein:

```
CHGDEVD DEV(Druckernamen) TRANSFORM(*YES) MFRTYPMDL(*HP5)
```

Eine Liste der verfügbaren Drucker sowie des entsprechenden Codes finden Sie im Hilfetext für den Parameter MFRTPMDL zum Befehl Create Printer Device Description (CRTDEVPRT) oder zum Befehl Change Device Description (CHGDEVD). Zu den Nachteilen von Host Print Transform gehört es, dass die Ausführung abhängig von der Komplexität der erstellten Ausgabe spürbar langsamer sein kann als das Senden des gleichen Dokuments zu einem IPDS/AFF-Drucker. Der Grund dafür ist, dass die Übertragung durch das Betriebssystem erfolgt und nicht direkt in die Hardware. Es gibt jedoch noch eine andere Lösung.

Für Drucker, die nicht AFP-fähig sind, sind mehrere Produkte verfügbar, mit denen AFP- und IPDS-Funktionalität hinzugefügt wird. Einige dieser Produkte erreichen das über eine Erweiterung mit einem Single In-line Memory Module (SIMM), das die zusätzliche Funktionalität bereitstellt.

Es sind weitere Produkte erhältlich, die Ihren Drucker nicht nur mit AFP erweitern, sondern außerdem als Printer Server agieren. Diese externe Anlage verbindet sich mit Ihrem LAN via Standardnetzwerkverkabelung über ein Standardparallelkabel. Wenn Sie mit einem PCL-Drucker arbeiten, versuchen sie am besten HPT zu verwenden. Wenn Leistungsprobleme auftreten, sollten Sie sich überlegen, ob es sich nicht lohnt, einen AFP SIMM- oder einen Printer Server anzuschaffen.

Schlüsselwörter

Verzeihen Sie das Wortspiel, aber der Schlüssel für die Erweiterung Ihrer Berichte mit AFP-Druckfunktionalität sind mehrere DDS-Druckerdatei-Schlüsselwörter. Eine Liste dieser Schlüsselwörter sehen Sie in Tabelle 4.1. Sie werden feststellen, dass diese Schlüsselwörter in solche für die Seitendefinition, Text- (oder Font-) Schlüsselwörter und Grafik-Schlüsselwörter gruppiert sind.

Tabelle 4.1: DDS Printer File Keywords für Advanced Function Printing

Schlüsselwort	Funktionstyp	Beschreibung
DUPLEX	Seitensteuerung	Ausdruck auf beiden Seiten der Seite.
ENDPAGE	Seitensteuerung	Auswerfen der aktuellen Seite nach dem Ausdruck des Berichts.
FORCE	Seitensteuerung	Ausgabe auf eine neue Seite erzwingen, wenn DUPLEX verwendet wird.
DOCIDXTAG	Seitensteuerung	Indexe zur Information für eine Seite festlegen.
ENDPAGGRP	Seitensteuerung	Beenden einer Seitengruppierung für die Dokumentenindexierung.
STRPAGGRP	Seitensteuerung	Starten einer Seitengruppierung für die Dokumentenindexierung.
CDEFNT	Text	Textfont mit kodiertem Fontnamen festlegen.
FNTCHRSET	Text	Textfont mit Fontzeichensatz festlegen.
FONT	Text	Textfont mit numerischer Font-ID festlegen.
POSITION	Text	Position für Text oder Grafik für die Anzeige auf der Seite festlegen.
TXTRTT	Text	Text drehen, um im angegebenen Winkel auszudrucken.

Schlüsselwort	Funktionstyp	Beschreibung
BOX	Graphik	Feld mit definierter Größe auf der Seite zeichnen.
GDF	Graphik	Inhalt einer Grafikdatendatei drucken.
LINE	Graphik	Zeile mit angegebener Länge und Ausrichtung auf der Seite definieren.
OVERLAY	Graphik	Overlay-Ressource gemeinsam mit Seitenausgabe festlegen.
PAGSEG	Graphik	Seitensegment-Ressource in angegebenem Bereich der Seite festlegen.

Schlüsselwörter für die Seitendefinition

In diesem Abschnitt erfahren Sie, welche Schlüsselwörter verwendet werden, um die Seite selbst zu steuern. Ganz am Anfang steht das Schlüsselwort DUPLEX. Dieses Schlüsselwort wird auf Datensatzebene verwendet, um zu definieren, ob der Druck auf beiden Seiten des Blatts auf einem Drucker erfolgen soll, der Duplex-Druck unterstützt. Das Schlüsselwort DUPLEX wird wie folgt angegeben:

```
DUPLEX(Duplexwert oder Program-to-System Feld)
```

Gültige Werte für dieses Schlüsselwort sind *NO (Ausdruck nur einer Seite des Blatts), *YES für den Ausdruck beider Seiten des Blatts in einem Format, das mit der linken Seite verknüpft wird, oder *TUMBLE für den Ausdruck beider Seiten in einem Format, das mit dem obersten Bereich verknüpft wird. Schauen Sie auf Abbildung 4.1, um eine grafische Darstellung dieser Optionen zu sehen.

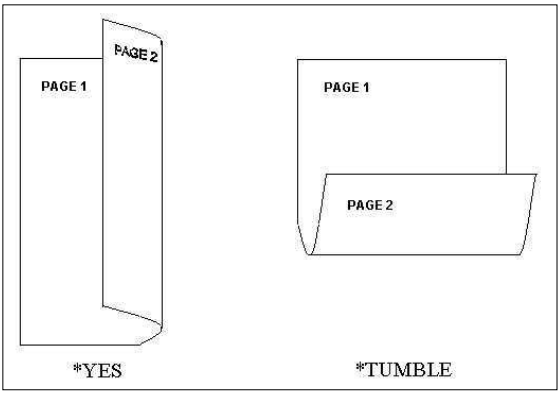


Abbildung 4.1: Optionen für das Schlüsselwort DUPLEX DDS.

Dieser Wert kann mit einem Program-to-System-Feld ersetzt werden, das mit einer Länge von 7 und dem Feldtyp A erstellt wird. Dieser Wert überschreibt den Wert, der im Parameter DUPLEX der Befehle Create Printer File (CRTPRTF), Change Printer File (CHGPRTF), oder Override Printer File (OVRPRTF) angegeben wird. Wenn dieses Schlüsselwort angegeben wird, ist es nur für das Datensatzformat aktiv, in dem es definiert wurde. Wenn das Schlüsselwort festgelegt wird, nachdem ein Feld oder eine Konstante für das aktuelle Datensatzformat angegeben wurde, wird das Schlüsselwort ignoriert. Das Schlüsselwort ENDPAGE sorgt dafür, dass die aktuelle Seite ausgegeben wird. Die gesamte Ausgabe für den aktuellen Datensatz wird vor der Ausgabe der Seite gedruckt. Dieses Schlüsselwort wird verwendet, wenn Sie eine Seite in einer bestimmten Situation abschließen möchten – z.B. nach dem Ausdruck von Gesamtsummen. Hier ein Beispiel für die Verwendung dieses Schlüsselworts:

```
A*
A*
A          R PAGTOT                    ENDPAGE
A                                      10'*PAGE TOTALS'
A*
```

In diesem Beispiel wird der Konstantentext *PAGE TOTALS ausgedruckt, bevor die Seite ausgegeben wird.

Wie bereits erwähnt, ist Duplex-Druck möglich, wenn entweder der Parameter DUPLEX im Befehl CHGPRTF, CRTPRTF, oder OVRPRTF angegeben wird, oder indem das Schlüsselwort DUPLEX DDS angegeben wird. Das DDS-Schlüsselwort FORCE erzwingt die Ausgabe der aktuellen Seite ohne Wechsel zur gegenüberliegenden Seite. Dieses Schlüsselwort wird ähnlich wie

das Schlüsselwort ENDPAGE verwendet. Das Schlüsselwort wird ignoriert, wenn der Duplex-Druck nicht aktiv ist. Der Datensatz, der das Schlüsselwort FORCE verwendet, wird ausgedruckt, bevor die Seite ausgegeben wird. Dieses Schlüsselwort wird wie folgt angegeben:

```
A*
A          R SKPSID                    FORCE
A                                      10'*END OF PAGE'
A*
```

Obwohl die nächsten drei Schlüsselwörter sich nicht auf die Seitensteuerung beziehen, werden sie auf Seitenebene definiert, wir haben sie daher unter der Seitensteuerungsüberschrift gruppiert. Das Schlüsselwort Document Index Tag (DOCIDXTAG) wird verwendet, um einen Index und einen Wert für die aktuelle Seite oder Seitengruppe zu definieren. Diese Werte können vom Client Access AFP Workbench Viewer verwendet werden, um spezifische Seiten innerhalb einer gespoolten Datei zu finden. Das Schlüsselwort DOCIDXTAG wird wie folgt angegeben:

```
DOCIDXTAG(Indexname, Indexwert, Indexlevel)
```

Alle Parameter dieses Schlüsselworts können entweder als Konstante oder als Program-to-System-Feld angegeben werden. Der Parameter Index Name definiert den Index selbst, den Sie hinzufügen oder erstellen möchten. Der Parameter „Indexwert“ definiert den Wert für den angegebenen Indexnamen. Die beiden ersten Parameter können jeder Zeichenwert von bis zu 250 Zeichen sein. Der dritte Parameter kann als PAGE angegeben werden, um den Indexwert auf die Seite anzuwenden, die das Schlüsselwort enthält oder um den Indexwert auf die Seitengruppe anzuwenden.

Wir befassen uns später mit Seitengruppen. Zuerst möchte ich Ihnen erklären, wie das Schlüsselwort DOCIDXTAG verwendet wird. Angenommen, Sie haben ein Programm entwickelt, das Kundenrechnungen erstellt. Wir könnten wie folgt ein Page-Level-Index-Tag erstellen:

```
A*
A          R INVHDR                DOCIDXTAG('Invoice, &INVN, PAGE)
A            INVN          10S 0P
A*
```

In diesem Beispiel wird der Wert des Felds INVN an den Page-Level-Index namens Invoice angehängt. Nachdem die Ausgabe erstellt wurde, kann der Benutzer nach einer bestimmten Rechnung suchen, indem er die gespoolte Datei im AFP Workbench Viewer öffnet und im Suche-Menü „Go To“ auswählt. Es wird eine Liste der erstellten Suchindexe oder Attribute angezeigt. Wenn Sie auf einen bestimmten Attributnamen klicken, wird außerdem eine Liste der verfügbaren Werte für dieses Attribut angezeigt. Abbildung 4.2 zeigt ein Beispiel für den Bildschirm des AFP Workbench Viewer „Go To“.

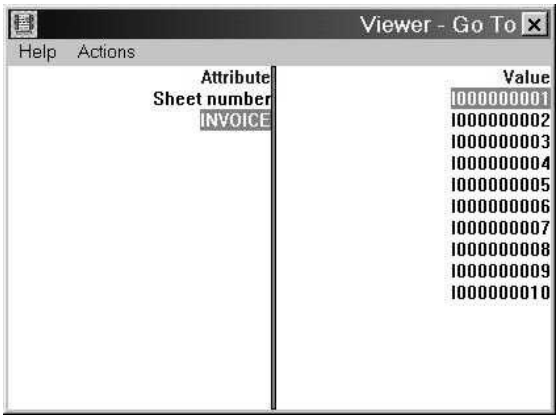


Abbildung 4.2: Der AFP Workbench Viewer kann verwendet werden, um nach DOCIDXTAG zu suchen.

Die Schlüsselwörter STRPAGGRP und ENDPAGGRP werden gemeinsam verwendet, um logische Seitengruppierungen für die Verwendung mit DOCIDXTAG zu definieren. Bei unserem früheren Beispiel nehmen wir an, dass die Rechnungen in Reihenfolge der Kundennummer erstellt wurden. Wir könnten ein Schlüsselwort STRPAGGRP in den Anfang der Gruppe einfügen, gefolgt vom Schlüsselwort DOCIDXTAG auf Gruppenebene, und anschließend ENDPAGGRP eingeben, wenn der erste Datensatz für einen neuen Kunden eingelesen wird. Das Schlüsselwort STRPAGGRP wird wie folgt angegeben:

```
STRPAGGRP(Gruppenname)
```

Der Parameter „Gruppenname“ wird verwendet, um der Gruppe einen Namen in dem Format einer Konstante oder eines Program-to-System-Felds hinzuzufügen. Das Schlüsselwort ENDPAGGRP hat keine Parameter. Abbildung 4.3 zeigt eine modifizierte Version unseres früheren Beispiels, mit der eine Kundennummerngruppe und ein -index zusätzlich zu unserem Kundennummernindex erstellt wird.

```
A*
A          R CUSTGRP                 STRPAGGRP('Kundennummer')
A                                    DOCIDXTAG('Kunde #', &CUSN, GROUP)
A            CUSN       10S 0P
A*
A          R INVHDR                  DOCIDXTAG('Rechnung', &INVN, PAGE)
A            INVN       10S 0P
A*
A          R CUSTENGP                ENDPAGGRP
A*
```

Abbildung 4.3: Dieses Beispiel erstellt Seiten- und Gruppen-Level-Indexe.

In diesem Beispiel erstellen wir außer dem Rechnungsattribut für Indexseiten in der Rechnungsnummer ein Kundennummernattribut für die Indizierung der Seiten für Kundennummerngruppen.

Text- und Font-Schlüsselwörter

Als Nächstes befassen wir uns mit text- und fontbezogenen Schlüsselwörtern. AFP bietet eine Reihe von Funktionen für Text, die es Ihnen erlauben, Text in vielen Fontstilen und Punktgrößen auszudrucken. Außerdem können Sie Text auf der Seite positionieren, ohne die Schlüsselwörter SPACE oder SKIP zu verwenden. Für die Definition von Fonts können drei Schlüsselwörter verwendet werden. Das Schlüsselwort CDEFNT wird verwendet, um den kodierten Font festzulegen, der für den Ausdruck einer beliebigen Konstante oder eines Felds verwendet wird. Dieses Schlüsselwort ist auf Feld- oder Datensatzebene gültig. Bei Definition auf Datensatzebene ist der kodierte Font für alle Felder und Konstanten im Datensatz gültig. Das Format für dieses Schlüsselwort ist wie folgt:

```
CDEFNT(Library-Namen/Coded Font Namen (*POINTSIZE Punkte))
```

Der Parameter Library-Namen ist optional; wenn er nicht definiert ist, lautet der Standardwert *LIBL. Der optionale Parameter *POINTSIZE erlaubt es Ihnen, die Punktgröße des ausgewählten Fonts auszuwählen. Coded Font definiert sowohl den Zeichensatz als auch die Codeseite für den ausgewählten kodierten Font. Verfügbare kodierte Fonts sind in der Bibliothek QFNTCPL enthalten, wobei der Objekttyp *CDEFNT lautet.

Das Schlüsselwort Font Character Set (FNTCHRSET) wird weitgehend ähnlich wie CDEFNT verwendet. Der Unterschied hier ist, dass das Schlüsselwort den Zeichensatz und die Codeseite separat definiert.

Das Schlüsselwort wird wie folgt definiert.

```
FNTCHRSET(Library/Font-Zeichensatz Library/Codeseite
          (*POINTSIZE Punkte))
```

Wie beim Schlüsselwort CDEFNT ist der Bibliotheksname optional. Wenn er nicht angegeben wird, lautet der Standardwert *LIBL. Sowohl der Wert für Font-Zeichensatz als auch die Codeseiten-Werte sind erforderliche Parameter. Der Punktgrößenparameter ist optional und muss als Sonderwert *POINTSIZ definiert werden, gefolgt von einem Wert zwischen.1 und 999.9 oder dem Sonderwert *NONE, der dazu führt, dass die Punktgröße auf Grundlage der Font-ID festgelegt wird. Verfügbare Font-Zeichensatz-Ressourcen und Codepage-Ressourcen finden Sie in QFNTCPL mit den Objekttypen *FNTCSET und *CDEPAG (respektive). Dieses Schlüsselwort kann auch auf Datensatzebene angegeben werden, um so alle Felder innerhalb eines Datensatzes zu konfigurieren, oder auf Feldebene für die Konfiguration eines einzigen Felds.

Zuletzt wird das Schlüsselwort FONT verwendet, um den Font mit Hilfe der numerischen Font-ID zu definieren. Dieses Schlüsselwort wird wie folgt angegeben:

```
FONT(Fontnummer (*POINTSIZE Punkte))
```

Eine Liste der Fontnummern für einige häufig verwendeten Fonts sehen Sie in Tabelle 4.2. Der optionale Punktgrößenparameter wird im gleichen Format definiert, das für die beiden Schlüsselwörter CDEFNT und FNTCHRSET gilt. Dieser Wert ist sowohl auf Datensatz- als auch auf Feldebene verfügbar.

Tabelle 4.2: Font-Nummern für beliebte Fonts			
Font #	**Fontname**	**Fonttyp**	**Größe**
11	Courier	Feste Laufweite	10 cpi
18	Courier Italic	Feste Laufweite	10 cpi
19	OCR-A	Feste Laufweite	10 cpi
20	Pica	Feste Laufweite	10 cpi
40	Gothic	Feste Laufweite	10 cpi
41	Roman Text	Feste Laufweite	10 cpi
760	Times Roman	Typograpisch	36 cpi
761	Times Roman Bold	Typographisch	18 cpi
762	Times Roman Bold	Typographisch	15 cpi
763	Times Roman Italic	Typographisch	18 cpi
764	Times Roman Bold Italic	Typographisch	21 cpi
765	Times Roman Bold Italic	Typographisch	18 cpi
2304	Helvetica Roman Bold	Typographisch	Skalierbar
2305	Helvetica Roman Italic	Typographisch	Skalierbar
2306	Helvetica Roman Italic Bold	Typographisch	Skalierbar
2308	Times New Roman Medium	Typographisch	Skalierbar
2309	Times New Roman Bold	Typographisch	Skalierbar
2310	Times New Roman Italic	Typographisch	Skalierbar
2311	Times New Roman Italic Bold	Typographisch	Skalierbar

Das Schlüsselwort POSITION wird verwendet, um die Position auf der ausgedruckten Seite anzugeben, in der eine bestimmte Konstante oder ein Feld beginnen soll. Dieses Schlüsselwort wird wie folgt verwendet:

```
POSITION(Vertikale Position, Horizontale Position)
```

Der Parameter „Vertikale Position“ definiert die Position abwärts vom Seitenanfang. Der Parameter „Horizontale Position“ definiert die Position von der linken Seite des Blatts. Die Werte für diese Parameter müssen der Maßeinheit entsprechen, die im Parameter UOM der Befehle CRTPRTF, CHGPRTF, oder OVRPRTF definiert ist. Es ist wichtig, zu wissen, welche Maßeinheit verwendet wird, wenn eine Datei erstellt wird, so dass Sie bei notwendigen Änderungen sicherstellen können, dass alle Schlüsselwörter für die Positionierung in der gleichen Maßeinheit spezifiziert werden.

Hier ein Beispiel für die Verwendung dieses Schlüsselworts:

```
A*
A            R HEADING                FONT(16951 (*POINTSIZE 36))
A              ITEM          15A 0P   POSITION(1.5, 3)
A*
```

Wenn wir diese Datei mit dem Wert UOM für *INCHES erstellen, druckt das Beispiel den Wert des Felds ITEM 1,5 Zoll vom Seitenrand entfernt sowie 3 Zoll vom linken Seitenrand aus. Der Feldwert wird mit dem Font Century Schoolbook mit einer Größe von 36 Punkten ausgedruckt.

Das letzte Textschlüsselwort, das Sie hier kennen lernen sollen, ist der Parameter TXTRTT. Dieses Schlüsselwort wird auf Feldebene angegeben und verwendet, um den Text des angegebenen Felds oder der Konstante in 90-Grad-Inkrementen von 0 aufwärts und einschließlich von 270 zu positionieren. Dieses Schlüsselwort wird wie folgt angegeben:

```
TXTRTT(Drehwinkel)
```

Grafik-Schlüsselwörter

Als Nächstes sehen wir uns die Grafik-Schlüsselwörter an, die es uns ermöglichen, auf unserer Seite zu „zeichnen“. Das Schlüsselwort BOX ermöglicht uns, auf unserer Seite Felder zu erstellen. Diese können verwendet werden, um aus unserer Anwendung heraus Formulare zu erstellen. Dieses Schlüsselwort wird auf Datensatzebene im folgenden Format angegeben:

```
BOX(Startpos. abwärts,Startpos. quer,Endpos. abwärts,Endpos.quer,Zeilenbreite)
```

Die beiden ersten Parameter werden verwendet, um die Ecke oben rechts des Felds zu definieren. Die beiden nächsten Parameter definieren die Ecke oben links des Felds. Der fünfte Parameter definiert die Dicke der Zeile, die für das Zeichnen des Felds verwendet wird. Der Parameter „Zeilenbreite“ kann entweder einen numerischen Wert zwischen ,001 Zoll und 22,75 Zoll oder einen von drei Sonderwerten enthalten: *NARROW, mit dem die Zeilenbreite mit ,008 Zoll festgelegt wird, *MEDIUM, mit dem die Linie mit 0,17 Zoll festgelegt wird und *WIDE, mit dem eine Zeilenbreite von 0,25 Zoll definiert wird.

Wie beim Schlüsselwort POSITION müssen die Werte, die für die Positionierungsparameter festgelegt wurden, in der gleichen Maßeinheit angegeben werden wie im Befehl CRTPRTF. Alle Parameter in diesem Befehl können von Program-to-System-Feldern ersetzt werden, die mit einer Länge von 5 und 3 Dezimalstellen definiert werden müssen. Wenn das Program-to-System-Feld für den Parameter Zeilenbreite verwendet wird, können keine Sonderwerte festgelegt werden. Angenommen, wir verwenden im Befehl CRTPRTF eine Maßeinheit *INCHES. In diesem Fall würde der

folgende Code ein 5-x-5-Zoll-Feld zeichnen, das sich eineinhalb Zoll vom oberen Rand der Seite entfernt sowie eineinhalb Zoll vom linken Rand entfernt befindet, wobei ein Zeilendurchschuss von 0,17 Inch verwendet wird.

```
A          R BOXRCD                BOX(1.5 1.5 5 5 *MEDIUM)
```

Das Schlüsselwort LINE funktioniert ähnlich wie das Schlüsselwort BOX. Es wird ebenfalls auf Datensatzebene im folgenden Format angegeben:

```
LINE (Startposition Abwärts, Startposition Quer, Zeilenlänge, Zeilenausrichtung,
      Line Weight, Line Pad (optional))
```

Die beiden ersten Parameter werden genau so angegeben wie für das Schlüsselwort BOX. Diese Werte definieren das Ende der Zeile. Der Parameter „Zeilenlänge“ definiert die Länge als Maßeinheit wie im Befehl Create Printer File – ab dem Startpunkt gesehen. Der nächste Parameter definiert die Richtung, in die die Zeile gezeichnet werden soll. Mit diesem Parameter werden die Sonderwerte *HRZ oder *VRT verwendet. *HRZ wird verwendet, um eine horizontale Zeile ab dem Startpunkt in der angegebenen Länge zu ziehen. Der Parameter Line Weight funktioniert wie der Parameter BOX. Zuletzt wird der Parameter Line Pad verwendet, um festzulegen, in welche Richtung die Zeilenbreite angewandt wird. Wenn wir z.B. eine horizontale Zeile über die Mitte der Seite mit der Zeilenweite 10 zeichnen möchten, definiert der Parameter Line Pad, auf welcher Seite der vertikale Startpunkt erweitert wird.

Gültige Werte für diesen Parameter sind:

- *TOP: Zeile über Startpunkt erweitern.
- *BOT: Zeile unterhalb Startpunkt erweitern.
- *LEFT: Zeile nach links vom Startpunkt erweitern.
- *RIGHT: Zeile nach rechts vom Startpunkt erweitern.

Die Optionen *TOP und *BOT sind nur gültig, wenn die Zeilenrichtung *HRZ lautet. Wenn die Zeilenrichtung *VRT lautet, können die Optionen *LEFT und *RIGHT für den Wert Line Pad verwendet werden.

Das Schlüsselwort GDF erlaubt es Ihnen, Daten von einer Grafikdatei aus auszudrucken. Eine *Grafikdatendatei* ist eine physische Datei, die Befehle enthält, die den IBM Graphic Object Content Architecture- (GOCA-) Spezifikationen entsprechen. GOCA ist ein Befehlssatz, der für die Erstellung von komplexen Grafiken verwendet wird. Dieser Befehlssatz ermöglicht es Ihnen, Zeilen, Bögen, Grafiken und sogar schattierte Bereiche zu erstellen. Der Grafikdatendatei-Datensatz selbst wird als ein Einbyte-Auftragscode für den Zeichenbefehl erstellt, gefolgt von einem Einbyte-Binärcode, der verwendet wird, um die zu beachtende Länge der Befehlsdaten anzugeben. Die folgenden Daten bestehen aus einer Gruppe von Parametern, die für die Definition der Attribute des zu zeichnenden Objekts verwendet werden. Zu diesen Attributen werden z.B. die Positionierung, der Zeilendurchschuss oder die Füllmuster gehören. Die Befehle werden in der physischen Quellenteildatei gespeichert und können später von unserer Anwendung mit dem Schlüsselwort GDF ausgegeben werden. Dieses Schlüsselwort selbst wird wie folgt verwendet:

GDF(GDF(Librarynamen/Dateinamen, Teildatei, oberste Position, rechte Position, Grafikhöhe, Grafikbreite, Grafikdrehung)

Alle Parameter mit Ausnahme des Bibliotheksparameters sind erforderlich und alle Parameter können entweder als Konstanten oder Program-to-System-Felder festgelegt werden. Der Parameter File Name kann optional den qualifizierten Bibliotheksnamen enthalten. Wenn der Bibliotheksname nicht angegeben wird, wird die Bibliotheksliste verwendet, die vorhanden ist, wenn die Ausgabe generiert wird. Die Positionierungsparameter werden auf die gleiche Weise verwendet wie im Schlüsselwort POSITION. Die Parameter für Höhe und Breite definieren die Größe der Grafik auf der Seite. Die Grafik wird anschließend auf die angegebene Größe skaliert. Jeder dieser Parameter akzeptiert Werte zwischen 0 und 22,75 Zoll. Der Rotationsparameter wird verwendet, um die Grafik auf der Seite zu drehen und kann in gültigen 90-Grad-Intervallen von 0 aufwärts einschließlich von 270 festgelegt werden. Tabelle 4.3 zeigt die Formatierung eines Zeilenbefehls in GOCA.

Tabelle 4.3: GOCA Datenstruktur für einen GLINE-Befehl

Position	Datentyp	Beschreibung
1	Auftragscode	GLINE Auftragscode (hex ‘81’)
2	Binär (ohne Vorzeichen)	Länge der folgenden Daten als Mehrfaches von 4
3–4	Binär (mit Vorzeichen)	Horizontale Startposition der Zeile (von hex 8000 bis hex 7FFFF)
5–6	Binär (mit Vorzeichen)	Vertikale Startposition der Zeile (von hex 8000 bis hex 7FFFF)
7–8	Binär (mit Vorzeichen)	Horizontale Endposition der Zeile (von hex 8000 bis hex 7FFFF)
9–10 (etc.)	Binär (mit Vorzeichen)	Vertikale Endposition der Zeile (von hex 8000 bis hex 7FFFF)

Bei diesem Beispiel können wir die Zeile zu zusätzlichen Endpunkten verlängern, indem wir einfach eine weitere horizontale und vertikale Position hinzufügen. Wichtig ist hier, dass der binäre Längenparameter der Länge des Positionsparameters entsprechen muss.

Das Schlüsselwort PAGSEG wird verwendet, um eine AFP-Seiten-Segment-Ressource gemeinsam mit den Daten zu zeichnen, die auf Ihrer Seite ausgedruckt werden. *Seitensegmente* verfügen über eine Vielzahl von Anwendungsmöglichkeiten einschließlich des Ausdrucks von Signaturen gemeinsam mit Ihrer Ausgabe. Das Schlüsselwort PAGSEG wird auf Datensatzebene definiert und wie folgt festgelegt:

PAGSEG(Library-Namen/Seitensegmentnamen, Position abwärts, Position quer, *SIZE Breite Höhe, *ROTATION Grad der Drehung)

Der Bibliotheksnamensbereich des Seitensegmentnamens ist optional. Wenn kein Bereich angegeben wird, wird standardmäßig die Bibliotheksliste verwendet, die beim Druck der Datei gültig war. Die Positionierungsparameter funktionieren auf die gleiche Weise wie die bereits beschriebenen Schlüsselwörter. Der optionale Parameter *SIZE wird verwendet, um das Seitensegment zu skalieren. Dieser Parameter wird mit Hilfe des Sonderwerts *SIZE festgelegt, gefolgt von der Höhe und der Breite. Diese Parameter werden in der Maßeinheit festgelegt, die in den Befehlen CRTPRTF oder OVRPRTF verwendet werden. Als Letztes wird der Parameter *ROTATION verwendet, um das Seitensegment in einem angegebenen Winkel von 0 bis 270 Grad in 90-Grad-Inkrementen zu drehen.

Ein entfernter Verwandter des Schlüsselworts PAGSEG ist das Schlüsselwort OVERLAY. Dieses Record-Level-Schlüsselwort wird verwendet, um eine AFP-Overlay-Ressource gemeinsam mit den gedruckten Daten auf Ihrer Seite zu generieren. Die primäre Verwendung der AFP-Overlays besteht darin, ein Formular gemeinsam mit Ihren Daten zu generieren, um vorgedruckte Formulare zu ersetzen. Dieses Schlüsselwort wird wie folgt kodiert:

```
OVERLAY((Library-Namen/Overlay-Namen, Position abwärts, Position quer,
          *ROTATION Grad der Drehung)
```

Die Parameter werden auf die gleiche Weise definiert wie im Schlüsselwort PAGSEG. Alle diese Parameter können entweder als Konstanten oder als Program-to-System-Felder spezifiziert werden.

AFP-Overlays sind ein wirkungsvolles Werkzeug, das problemlos erstellt werden kann. Sie können nämlich mit Hilfe eines AFP-Druckertreibers für Windows von IBM einen Overlay mit einem beliebigen Windows-Programms erstellen, das druckfähig ist. Als Nächstes befassen wir uns mit dem Prozess der Erstellung eines Overlays in Microsoft Word, wobei wir das resultierende Overlay in einer Anwendung mit Hilfe des Schlüsselworts OVERLAY verwenden.

Wie bereits erwähnt, bildet der Windows AFP-Druckertreiber die Grundlage unseres Overlay-Puzzles. Dieser ist in Client Access Express enthalten oder kann von der IBM Printer-Webseite (*www.ibm.com/printers*) heruntergeladen werden. Sobald Sie den Treiber auf Ihrem System gespeichert haben, müssen Sie einen neuen Windows-Drucker erstellen, der diesen Treiber verwendet. Hierzu wechseln Sie in den Druckerordner Ihres Computers (Startmenü/Einstellungen/Drucker). Doppelklicken Sie auf „Drucker hinzufügen“.

Wenn der Assistent für das Hinzufügen neuer Drucker erscheint, wählen Sie „lokaler Drucker“. Wählen Sie aus der Liste der angezeigten Drucker „IBM“. Blättern Sie in der Liste, bis Sie die Drucker sehen, die in Abbildung 4.4 aufgeführt werden.

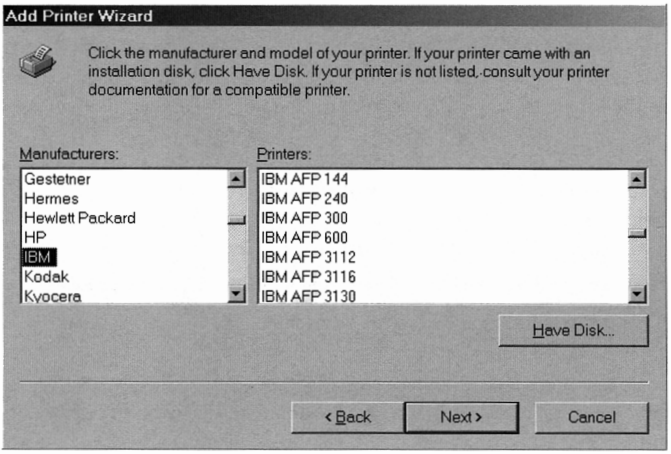


Abbildung 4.4: Der AFP-Druckertreiber wird mit dem Add Printer Wizard hinzugefügt.

Wenn Sie planen, die Ausgabe zu einem bestimmten IBM AFP-Druckermodell zu senden, wählen Sie dieses aus der Liste aus. Wenn nicht oder wenn Sie planen, die Ausgabe zu einem PCL-Drucker mit Hilfe von Host Print Transform zu senden, verwenden Sie einen der ersten vier Drucker in der Liste. Bei diesen vier Treibern informiert die angezeigte Zahl – z.B. die 144 für IBM AFP 144 – über die Anzahl der Dots per Inch (DPI). Wählen Sie den Drucker aus, der der DPI-Auflösung Ihres Druckers entspricht. Sobald Sie Ihren Drucker ausgewählt haben, klicken Sie auf „Weiter“. Wählen Sie in dem Bildschirm, der in Abbildung 4.5 gezeigt wird, FILE: (Datei) als Druckeranschluss aus.

Damit wird es möglich, die Ausgabe von unserem AFP-Treiber in eine Datei anstatt direkt zu einem Drucker zu senden. Klicken Sie auf Next (Weiter) und vergeben Sie für diesen Drucker einen Namen. Sinnvoll ist ein Namen wie z.B. „AFP OVERLAY DRIVER“, so dass Sie später sofort erkennen, für welchen Zweck Sie diesen Drucker erstellt haben. Wenn der Assistent für das Hinzufügen neuer Drucker Sie fragt, ob Sie eine Testseite ausdrucken möchten, antworten Sie mit „Nein“, da die Ausgabe sowieso nur in die Datei gesendet wird und weil es keine einfache Methode gibt, um diese Datei an diesem Punkt zu einem Drucker zu senden. Nun wechseln Sie zurück zum Druckerordner und klicken mit der rechten Maustaste auf den neu erstellten AFP-OVERLAY-DRIVER-Drucker. Wählen Sie aus dem Kontextmenü „Eigenschaften“. Wenn das Dialogfeld „Druckereigenschaften“ angezeigt wird, wählen Sie das Register „Papier“. Ändern Sie nun den Ausgabetyp zu Overlay – wie in Abbildung 4.6 gezeigt.

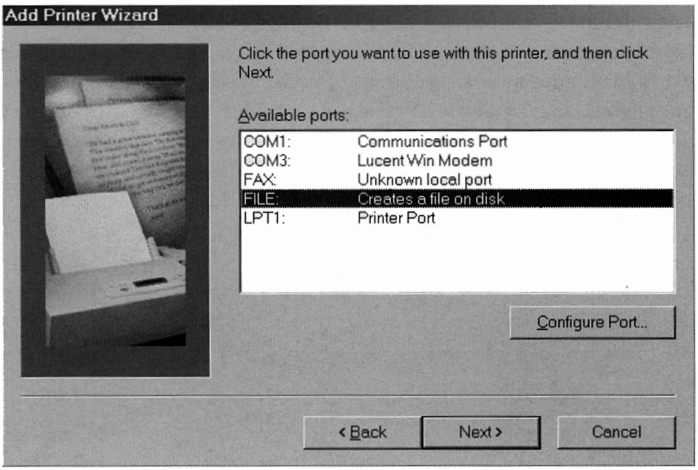


Abbildung 4.5: Wählen Sie den FILE-Anschluss, um die Ausgabe des AFP-Treibers in eine Datei zu senden.

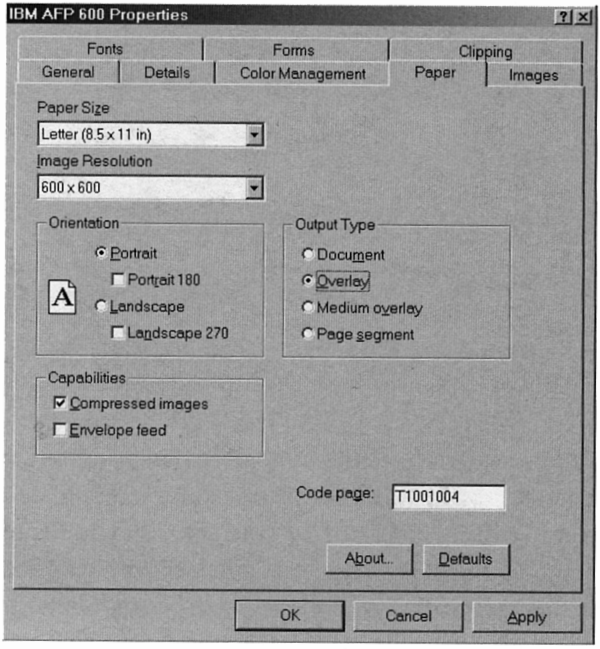


Abbildung 4.6: Ausgabetyp in Overlay ändern.

Damit wird sichergestellt, dass die erzeugte Ausgabe immer das Overlay-Format aufweist. Beachten Sie, dass die gleichen Instruktionen mit wenigen Ausnahmen ebenfalls ein Seitensegment erstellen können.

Wählen Sie dazu im Dialogfeld „Druckereigenschaften“ das Register „Schriften“. Deaktivieren Sie in dem Bildschirm, der in Abbildung 4.7 gezeigt wird, die Option Use Outline Fonts.

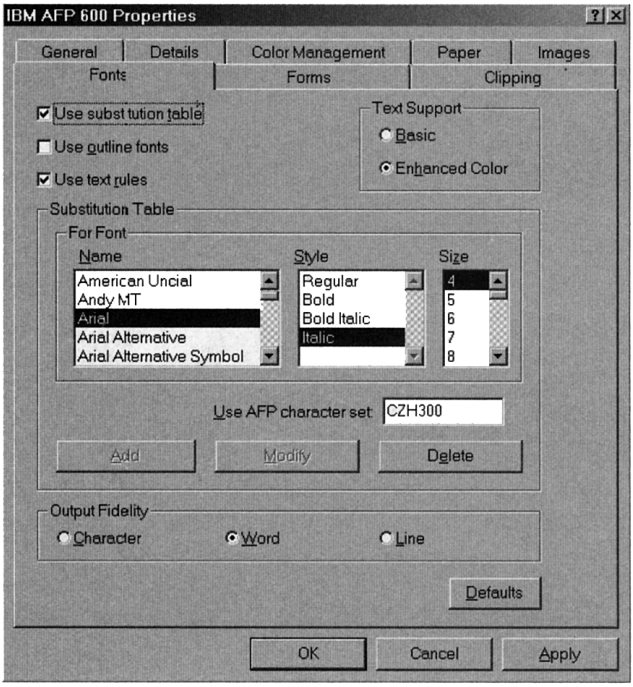


Abbildung 4.7: Das Font-Register wird verwendet, um zu kontrollieren, wie Fonts im AFP-Treiber übersetzt werden.

Mit diesen beiden Schritten werden einige häufig auftretende Probleme vermieden, die beim Versuch, Overlay zu generieren, vorkommen können. Wenn Sie versuchen, Ausgaben zu verwenden, deren Ausgabetyp nicht auf Overlay eingestellt wurde, tritt ein Fehler auf, sobald Sie beginnen, die Overlay-Ressource auf der 400 zu

erstellen. Außerdem passiert es sehr oft, wenn Sie einen Overlay auf einem Drucker ausdrucken, der nicht AFP-fähig ist, wie z.B. ein HP LaserJet, dass Ihre Schriften nicht einwandfrei angezeigt werden, wenn die Option „Outline-Fonts verwenden“ aktiviert ist. Klicken Sie auf „OK“, um die Einstellungen zu speichern Jetzt müssen wir noch einige Konfigurationsarbeiten auf der 400 erledigen. Als Erstes müssen wir auf der 400 einen freigegebenen Ordner erstellen, der verwendet werden wird, um die Overlay-Ressourcen an die 400 zu übertragen. Wechseln Sie zur Befehlseingabe und geben Sie folgenden Befehl ein:

```
CRTFLR FLR(OVERLAYS)
```

Nun erstellen wir eine Datei, von der der Overlay gespeichert wird, und erstellen die Ressource für den Overlay. Diese Datei muss eine Datensatzlänge von 1021 Byte aufweisen. Um diese Datei zu erstellen, verwenden wir den Befehl Create Physical File (CRTPF) wie folgt:

```
CRTPF FILE(QGPL/OVERLAYS) RCDLEN(1021)
```

Rechnungserstellung

Jetzt haben Sie alle erforderlichen Einstellungen vorgenommen, und es ist an der Zeit, das Formular zu erstellen. Wir werden dabei alle Schritte abarbeiten, die benötigt werden, um Microsoft Word zu verwenden, um ein einfaches Rechnungsformular zu erstellen. Als Erstes erstellen wir ein neues leeres Dokument in Word. Hierzu wechseln wir in das Menü „Datei" und wählen „Neu". Nun wählen Sie die leere Vorlage und klicken „OK". Als Nächstes wechseln Sie erneut zum Dateimenü und wählen „Drucken". Wir müssen Word mitteilen, dass wir unseren AFP-Druckertreiber verwenden, so dass das von uns erstellte Dokument für den AFP-Treiber formatiert werden wird.

Wenn der Druckdialog angezeigt wird, öffnen Sie die Liste der Drukker und wählen AFP Overlay Printer. Nun klicken Sie einfach auf „Schließen", um das Dialogfeld zu schließen. Als Nächstes wechseln Sie in das Dateimenü und wählen „Seite einrichten". Nun reduzieren Sie alle Ränder auf 0. Wenn Sie auf „OK" klicken, informiert Sie Word, dass die Ränder sich außerhalb des druckbaren Bereichs der Seite befinden. Klicken Sie auf „Ändern", um Word die Seitenränder automatisch auf die Mindestgröße anpassen zu lassen, die vom Druckertreiber mindestens benötigt werden. Damit erhält Ihr Formular den größtmöglichen Bereich. Nun beginnen wir mit dem Formular selbst. Als Erstes verschieben Sie Ihren Mauszeiger ganz zum Seitenanfang und klicken in der Schaltflächenleiste auf „Mittig". In diesem Symbol, das in Abbildung 4.8 gezeigt wird, werden mehrere Textzeilen mittig auf einer Seite ausgerichtet angezeigt.

Abbildung 4.8: Verwenden Sie dieses Symbol, um den Text in Ihrem Dokument zu zentrieren.

Nun klicken Sie auf das Formatmenü und klicken Sie auf „Schriftart“. Wählen Sie Arial 28 mit Fettdruck und klicken Sie auf OK. Nun tippen Sie das Wort „Rechnung“ ein. Als Nächstes reduzieren Sie mit Hilfe des Punktgrößen-Auswahlreglers in der Schaltflächenleiste die Schriftgröße auf 18 Punkte und deaktivieren den Fettdruck, indem Sie auf das entsprechende Symbol in der Schaltflächenleiste klicken. Nun geben Sie den Namen und die Adresse Ihres Unternehmens (ABC Corp., 123 Main Street, Anytown, US 99999 wie in unserem Beispiel). Als Nächstes erstellen wir eine Tabelle. Hierzu wechseln Sie in das Menü „Tabelle“ und wählen „Tabelle einfügen“. Wenn das Dialogfeld „Tabelle einfügen“ erscheint, wählen Sie zwei Zeilen und zwei Spalten (d.h. den Standardwert) und klicken auf „OK“. Damit wird die neue Tabelle in Ihre Seite mit zwei Spalten, die jeweils die Hälfte der Seite einnehmen, eingefügt und jeweils zwei Zeilen, die jeweils ungefähr einen Buchstaben hoch sind. Markieren Sie die beiden untersten Tabellenzellen, indem Sie auf die linke Zelle klicken und zur rechten Zelle herüberziehen. Nun klicken Sie auf das Menü „Tabelle“ und wählen „Zellenhöhe und Zellenbreite“. Ändern Sie in der Registerkarte Rows das Listenfeld rechts von „Automatisch“ zu „Genau“ und ändern Sie die Zellenhöhe auf zwei Zoll (2”). Diese Tabelle wird für die Abschnitte „Ship to“ (Liefern an) und „Bill to“ (Rechnung an) des Formulars verwendet. Klicken Sie auf die oberste Zelle links und tippen Sie „Ship to“ (Liefern an) ein. Als Nächstes klicken Sie auf die Zelle oben rechts und geben „Bill to“ (Rechnung an) ein. An diesem Punkt sollte Ihr Dokument ähnlich wie in Abbildung 4.9 aussehen.

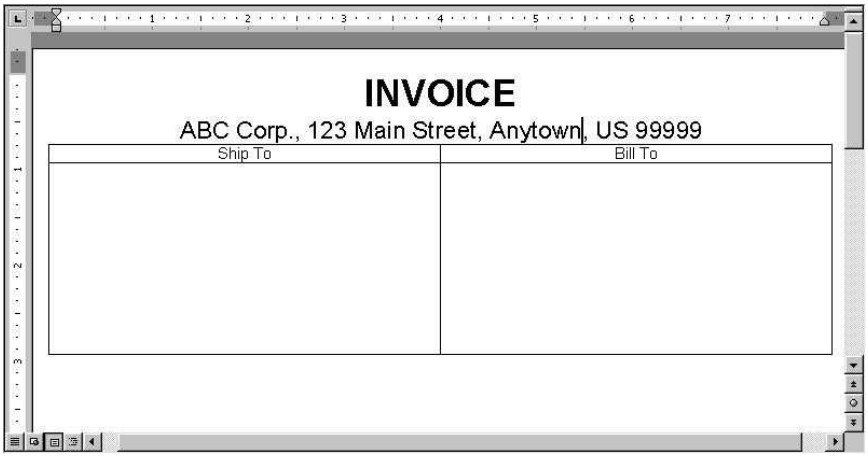


Abbildung 4.9: Dies ist der Kopfbereich unserer Rechnung in Microsoft Word.

Nun erstellen wir eine zweite Tabelle, die verwendet wird, um die Detailzeilen in unserer Rechnung zu zeigen. Wechseln Sie erneut in das Menü „Tabelle“ und wählen Sie erneut „Tabelle einfügen“. Diesmal erstellen Sie eine Tabelle mit vier Spalten und zwei Reihen. Nun geben Sie von links nach rechts in die Zellen in der obersten Zeile Item, Quantity, Price und Amount ein (Position, Menge, Preis und Betrag). Markieren Sie auch hier die zweite Zeile und ändern Sie diesmal die Zeilenhöhe auf sechsundeinhalb Zoll (6.5”). Dieser Abschnitt des Dokuments sollte der Abbildung 4.10 ähneln.

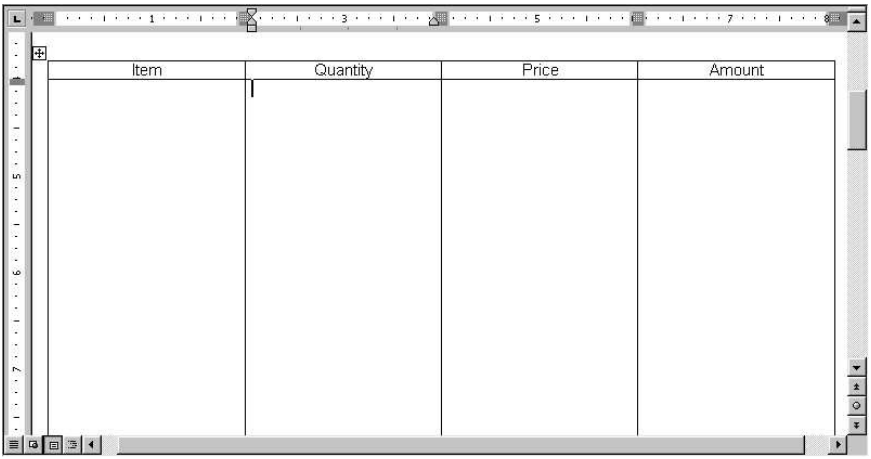


Abbildung 4.10: Der Detailabschnitt unserer Rechnung wird mit einer Tabelle in MS Word erstellt.

Nun wechseln wir in den Bereich ganz unten auf der Seite und verwenden den 12-Punkt-Font Arial, fügen acht Tab-Stopps ein und geben anschließend den Text „Rechnung Total“ ein. Speichern Sie das Dokument, indem Sie in das Dateimenü wechseln und „Speichern“ wählen. Es ist eine gute Idee, das Worddokument in den freigegebenen Ordner auf Ihrer iSeries oder AS/400 zu speichern. Hierzu klicken Sie im Speicherdialog auf das Listenfeld „Speichern in“ und wählen „Netzwerk“. Nun suchen Sie nach dem Systemnamen Ihrer 400 und doppelklicken darauf. Dann suchen Sie nach dem QDLS-Ordner, der alle Ihre freigegebenen Ordnerobjekte enthält. Als Nächstes suchen Sie nach dem Overlays-Ordner, den wir zuvor erstellt haben. Das fertige Dokument sollte wie das Dokument in Abbildung 4.11 aussehen.

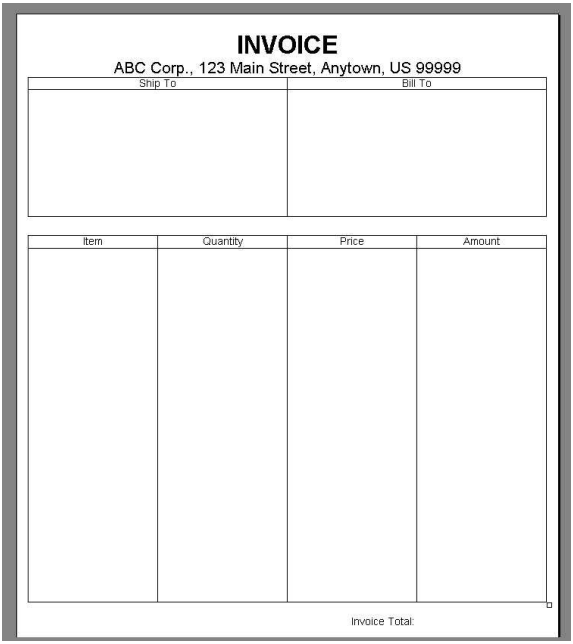


Abbildung 4.11: Das fertige Rechnungsformular

Nun klicken Sie auf das Menü „Datei“ und wählen „Drucken“. Prüfen Sie, ob der AFP Overlay-Drucker immer noch als der Drukker für das Dokument aufgeführt wird, und klicken Sie auf „OK“. Wenn Sie dazu aufgefordert werden, den Dateinamen und den Speicherort anzugeben, wählen Sie den Overlays-Ordner auf die gleiche Weise wie zuvor und geben als Dateinamen Invoice.prn ein. Nun haben Sie die erforderlichen Vorbereitungen auf dem PC getroffen. Den Rest der Arbeit erledigen Sie auf der 400.

Die Erstellung der Overlay-Ressource aus der Ausgabe erfolgt in zwei Schritten. Als Erstes müssen wir die Ausgabe unseres AFP-

Treibers aus dem freigegebenen Ordner in die zuvor erstellte Druckerdatei OVERLAY kopieren. Hierzu verwenden wir den folgenden Befehl „Copy from PC Document (CPYFRMPDC)“:

```
CPYFRMPCD FROMFLR(OVERLAYS) TOFILE(QGPL/OVERLAYS)
          FROMDOC(INVOICE.PRN) TOMBR(INVOICE) TRNTBL(*NONE)
          TRNFMT(*NOTEXT)
```

Jetzt kann die Overlay-Ressource erstellt werden. Hierzu geben Sie den Befehl Create Overlay (CRTOVL) wie folgt ein:

```
CRTOVL OVL(QGPL/INVOICE) FILE(QGPL/OVERLAYS)
```

Nun müssen wir nur noch unsere Druckerdatei so modifizieren, dass sie das Schlüsselwort OVERLAY enthält. Das RPG-Programm in Abbildung 4.12 zeigt Ihnen, wie das Ganze funktioniert.

```
 ************************************************************************
 *
 * Kompilierung:       CRTBNDRPG PGM(libname/INV001RG)
 *                               SRCFILE(libname/QRPGLESRC)
 *                               SRCMBR(INV001RG)
 ************************************************************************
FINV001P1  O   E              PRINTER OFLIND(*IN01)
C*
C* OUTPUT THE OVERLAY
C                   Write     Overlay
C*
C* WRITE INVOICE HEADER
C                   Eval      SHIP1=*ALL'X'
C                   Eval      BILL1=*ALL'X'
C                   Eval      SHIP2=*ALL'X'
C                   Eval      BILL2=*ALL'X'
C                   Eval      SHIP3=*ALL'X'
C                   Eval      BILL3=*ALL'X'
```

Abbildung 4.12: Verwenden Sie dieses ILE-RPGProgramm für die Erstellung der Beispielausgabe für unser Overlay (Teil 1 von 2).

```
C                   Eval      SHIP4=*ALL'X'
C                   Eval      BILL4=*ALL'X'
C                   Eval      SHIP5=*ALL'X'
C                   Eval      BILL5=*ALL'X'
C                   Write     Header
C*
C* WRITE DETAIL LINES
C                   Do        34            X                 2 0
C                   Eval      Item='ITEM'+%EDITC(X:'4')
C                   Eval      QTY=X
C                   Eval      PRICE=X/34
C                   Eval      Amount=Qty*Price
C                   Write     Detail
C                   Eval      TotAmt=TotAmt+Amount
C                   EndDo
C*
C* OUTPUT TOTAL LINE AND END
C*
C                   Write     Total
C                   Eval      *INLR=*ON
```

Abbildung 4.12: Verwenden Sie dieses ILE-RPGProgramm für die Erstellung der Beispielausgabe für unser Overlay (Teil 2 von 2).

Dieses Programm gibt einige Beispieldaten aus, die für unser Formular geeignet sind. Die Druckerdatei-DDS, die in Abbildung 4.13 gezeigt wird, enthält das Schlüsselwort OVERLAY und wird verwendet, um unsere fertige Ausgabe zu erstellen.

```
*****************************************************************************
*  Kompilierung:     CRTPRTF FILE(library/INV001P1)
*                            SRCFILE(library/QDDSSRC)
*                            DEVTYPE(*AFPDS)
*                            PAGRTT(0)
*                            PAGESIZE(66 80)
A*****************************************************************************
A*
A      R OVERLAY                          OVERLAY(INVOICE 0 0)
A      R HEADER
A        SHIP1              30          3SKIPB(7)
A        BILL1              30         43SPACEA(1)
A        SHIP2              30          3
A        BILL2              30         43SPACEA(1)
A        SHIP3              30          3
A        BILL3              30         43SPACEA(1)
A        SHIP4              30          3
A        BILL4              30         43SPACEA(1)
A        SHIP5              30          3
A        BILL5              30         43SKIPA(23)
A      R DETAIL
A      ITEM                 10           3
A      QTY                  10  0      20EDTCDE(4)
A      PRICE                13  5      45EDTCDE(M)
A      AMOUNT               13  2      65EDTCDE(M)
A                                        SPACEA(1)
A      R TOTAL                           SKIPB(62)
A      TOTAMT               15  2      63EDTCDE(M)
```

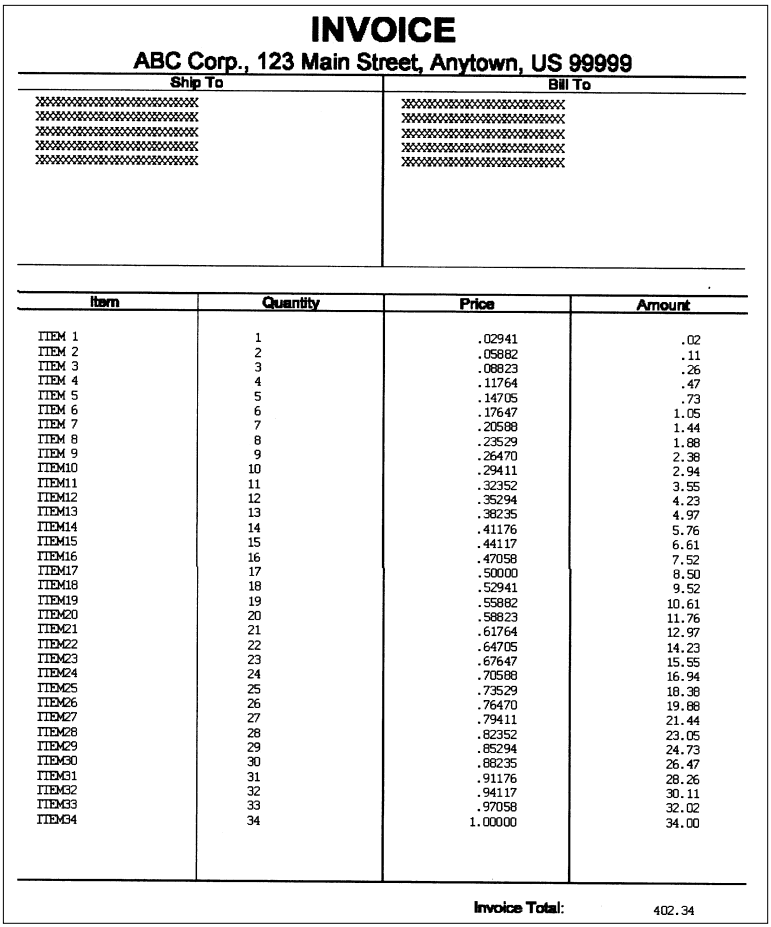
Abbildung 4.13: Diese Druckerdatei verwendet das Schlüsselwort OVERLAY für die Erstellung unserer fertigen Rechnung.

Sie werden bemerken, dass das Schlüsselwort OVERLAY einen eigenen Datensatz aufweist. Der Grund dafür ist, dass alle Felder, die im gleichen Datensatz ausgedruckt werden, mit dem Schlüsselwort POSITION positioniert werden. In diesem Fall positionieren wir die Felder mit Hilfe der Schlüsselwörter SPACEX und SKIPX.

Nachdem Sie diese beiden Objekte kompiliert haben, rufen Sie das Beispielprogramm auf, indem Sie Folgendes eingeben:

```
CALL PGM(INV001RG)
```

Die Ausgabe, die generiert wird, sollte ähnlich wie in Abbildung 4.14 aussehen.



INVOICE

ABC Corp., 123 Main Street, Anytown, US 99999

Ship To	Bill To
XXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXX

Item	Quantity	Price	Amount
ITEM 1	1	.02941	.02
ITEM 2	2	.05882	.11
ITEM 3	3	.08823	.26
ITEM 4	4	.11764	.47
ITEM 5	5	.14705	.73
ITEM 6	6	.17647	1.05
ITEM 7	7	.20588	1.44
ITEM 8	8	.23529	1.88
ITEM 9	9	.26470	2.38
ITEM10	10	.29411	2.94
ITEM11	11	.32352	3.55
ITEM12	12	.35294	4.23
ITEM13	13	.38235	4.97
ITEM14	14	.41176	5.76
ITEM15	15	.44117	6.61
ITEM16	16	.47058	7.52
ITEM17	17	.50000	8.50
ITEM18	18	.52941	9.52
ITEM19	19	.55882	10.61
ITEM20	20	.58823	11.76
ITEM21	21	.61764	12.97
ITEM22	22	.64705	14.23
ITEM23	23	.67647	15.55
ITEM24	24	.70588	16.94
ITEM25	25	.73529	18.38
ITEM26	26	.76470	19.88
ITEM27	27	.79411	21.44
ITEM28	28	.82352	23.05
ITEM29	29	.85294	24.73
ITEM30	30	.88235	26.47
ITEM31	31	.91176	28.26
ITEM32	32	.94117	30.11
ITEM33	33	.97058	32.02
ITEM34	34	1.00000	34.00

Invoice Total: 402.34

Abbildung 4.14: Unsere fertige Ausgabe mit Overlay.

Befassen wir uns mit einem Beispiel in der Praxis, bei dem wir einige der anderen Schlüsselwörter verwenden, die Sie in diesem Buch kennen gelernt haben. Hierzu erstellen wir eine Anwendung, die verwendet werden kann, um ein einfaches Balkendiagramm mit Hilfe der bereitgestellten Informationen zu generieren. Unser Beispielprogramm wird mehrere der AFP-Schlüsselwörter verwenden, die wir in diesem Kapitel kennen gelernt haben – einschließlich von BOX, LINE, PAGRTT, TXTRTT, und POSITION. Die Druckerdatei für unser Balkendiagramm sehen Sie in Abbildung 4.15.

```
 *
 * Befehl erstellen: CRTPRTF FILE(lib/BGR001P1)
 *                   SRCFILE(lib/QDDSSRC)
 *                   DEVTYPE(*AFPDS)
 *
 * Grafikbereich zeichnen
 *
A         R GRAPH                PAGRTT(90)
A                                LINE(0.1 1.5 6 *VRT *MEDIUM)
A                                LINE(6.25 1.5 9 *HRZ *MEDIUM)
A           TITLE    30A         FONT(16951 (*POINTSIZE 36))
A                                POSITION(.5 2.5)
*
* Leiste zeichnen
*
A         R DRAWBAR
A                                BOX(&STRY &STRX &ENDY &ENDX    +
A                                *MEDIUM)
A           STRY      5S 3P
A           STRX      5S 3P
A           ENDY      5S 3P
A           ENDX      5S 3P
*
* HASH-Zeilen und Werte zeichnen
*
A         R HRZLINE
A                                LINE(&Y .1 1.3 *HRZ *NARROW)
A           VALUE    10A         FONT(16951 (*POINTSIZE 10))
A                                POSITION(&Y1 .5)
```

Abbildung 4.15: Diese Druckerdatei wird verwendet, um ein einfaches Balkendiagramm zu erstellen (Teil 1 von 2).

```
A           Y         5S 3P
A           Y1        5S 3P
*
* Leistentext zeichnen (Rotation 90 Grad)
*
A         R BARTEXT
A           TEXT     10A         TXTRTT(90)
A                                FONT(2304 (*POINTSIZE 12))
A                                POSITION(6.4 &X)
A           VALU      6  0       POSITION(&Y &X)
A                                FONT(2304 (*POINTSIZE 12))
A           Y         5S 3P
A           X         5S 3P
*
* Aktuelle Seite auswerfen
*
A         R PAGEND               ENDPAGE
```

Abbildung 4.15: Diese Druckerdatei wird verwendet, um ein einfaches Balkendiagramm zu erstellen (Teil 2 von 2).

Der erste Datensatz in der Datei mit der Bezeichnung GRAPH wird verwendet, um den Haupt-Grafikbereich gemeinsam mit dem Titel für das Balkendiagramm zu zeichnen. Das nächste Datensatzformat mit dem Namen DRAWBAR wird angegeben, um die einzelnen Balken für unser Diagramm zu zeichnen. Es wird Ihnen sicher auffallen, dass wir Program-to-System-Felder gemeinsam mit dem Schlüsselwort BOX verwenden, um die Position und Größe jeder Leiste zu definieren. Damit erhalten wir maximale Flexibilität für die Definition der Balken in unserem Diagramm. Das Datensatzformat mit dem Namen HRZLINE wird verwendet, um die „Hash-Linien“ für unser Diagramm zu erstellen und um die Werte für diese Hash-Zeichen zu generieren. Das nächste Datensatzformat mit der Bezeichnung BARTEXT, wird verwendet, um sowohl die Überschrift für die einzelnen Balken im Diagramm zu generieren als auch den Wert für den Balken, der direkt

über dem obersten Bereich jedes Balkens ausgedruckt wird. Das endgültige Datensatzformat heißt PAGEND und wird gemeinsam mit dem Schlüsselwort ENDPAG verwendet, um die aktuelle Seite nach dem Ausdruck auszugeben. Wenn Sie den Befehl Create Printer File (CRTPRTF) für diese Datei eingeben, achten Sie darauf, die Option *AFPDS für den Druckergerätetyp-Parameter wie folgt anzugeben:

```
CRTPRTF FILE(lib/BGR001P1) SRCFILE(lib/QDDSSRC) DEVTYPE(*AFPDS)
```

Nun haben wir die Druckerdatei erstellt und können uns als Nächstes mit der ILE RPG-Programm-Source in Abbildung 4.16 beschäftigen.

```
 *
 *
 *  Kompilierbefehl: CRTBNDRPG PGM(lib/BGR001RG)
 *                              SRCFILE(lib/QRPGLESRC)
 *                              SRCMBR(BGR001RG)
 *
FBGR001P1  O   E             PRINTER
DVal             S              6    DIM(12)
DTxt             S             10    DIM(12)
DHRZINCR         S              5  3
DnumV            S              6  0
Ddiff            S              6  0
DIx              S              6  0
Dmin             S              6  0
Dmax             S              6  0
C     *ENTRY       PList
C                  Parm                    Title           30
C                  Parm                    MinVal          15 5
C                  Parm                    MaxVal          15 5
C                  Parm                    NumVals         15 5
C                  Parm                    QKl
C                  KKrm                    Txr·
```

Abbildung 4.16: Dieses Programm erstellt ein einfaches Balkendiagramm unter Verwendung von AFP-Schlüsselwörtern (Teil 1 von 2).

```
 * Ausgabe des Diagrammrahmens
                  Write      GRAPH
 * Hash-Zeilen ausschreiben
C                 Eval       Diff=(MaxVal-MinVal)/8
C                 Eval       Y=6.25
C                 Eval       Min=MinVal
C                 Eval       Max=MaxVal
C                 For        Ix=Min BY Diff TO Max
C                 Eval       Value=%EDITC(Ix:'L')
C                 Eval       Y1=Y-.1
C                 Write      HRZLINE
C                 Eval       Y=Y-.583
C                 EndFor
*  Ausgabe-Leiste
C                 Eval       HrzIncr=8.5/NumVals
C                 Eval       Strx=2
C                 Eval       EndY=6.25
C                 Eval       NumV=NumVals
C                 For        Ix=1 TO NumV
C                 Eval       EndX=StrX+HrzIncr*.75
C                 Move       Val(Ix)      Valu
C                 Eval       StrY=6.25-(5.83*(Valu/(Max-Min)))
C                 Eval       Y=StrY-.1
C                 Eval       X=StrX
C                 Write      DRAWBAR
C                 Eval       Text=Txt(Ix)
C                 Write      BARTEXT
C                 Eval       Strx=Strx+HrzIncr
C                 EndFor
*  Seite auswerfen
C                 Write      PAGEND
C                 Eval       *INLR=*ON
C                 Return
```

Abbildung 4.16: Dieses Programm erstellt ein einfaches Balkendiagramm unter Verwendung von AFP-Schlüsselwörtern (Teil 2 von 2).

Die Ausgabe dieses Programms erfolgt in unsere Beispiel-Diagramm-Druckerdatei. Das Programm akzeptiert sechs Parameter. Diese sind:

- Titel (alphanumerisch, 30 Zeichen),
- Minimaler Diagrammwert (15,5 numerisch; weniger als 999999),
- Maximaler Diagrammwert (15,5 numerisch),
- Anzahl der Diagrammwerte (15,5 numerisch; weniger als 999999),
- Diagrammwerte (Array mit 12 6-Zeichen 0 Dezimalstellenfelder),
- Balkentext (Array mit 12 alphanumerischen 10-Zeichen-Feldern).

Nachdem Sie das GRAPH-Datensatzformat ausgegeben haben, bestimmt das Programm das numerische Inkrement, das für die Hash-Zeichenwerte verwendet werden soll. Dies geschieht, indem der Mindestwert für das Diagramm vom Maximalwert für das Diagramm abgezogen wird und die Differenz durch 10 dividiert wird (d.h. durch die Anzahl der zu verwendenden Hash-Zeichen). Diese Zahl wird als Inkrement für die Hash-Zeichen verwendet. Als Nächstes verwenden wir eine FOR/ENDFOR-Schleife, die den Mindestdiagrammwert und dann den Maximaldiagrammwert mit dem zuvor festgestellten Inkrement abarbeitet. Diese Zahl und eine Zeile werden an einer Position auf der Seite beginnend bei 6,25 Zoll abwärts ausgegeben und verringern sich mit jeder Zeile um 583 Zoll. Sobald alle Hash-Zeilen ausgegeben wurden, müssen wir die Balken und den Text für die Diagrammwerte ausgeben.

Wir legen die Breite jedes Balkens fest, indem wir 8,5 dividieren, also den Wert der für die Diagrammbreite verfügbaren Inches geteilt durch die Anzahl der anzuzeigenden Werte. Dieser horizon-

tale Inkrementwert wird verwendet, um die Start- und Endpositionen X für den Balken festzulegen. Wir bestimmen die Startposition Y für den Balken, indem wir den Balkenwert durch die Differenz zwischen Minimal- und Maximalwerten teilen. So erfahren wir, welcher Prozentsatz der vollständigen Diagrammhöhe dieser Balken aufweisen soll. Wir multiplizieren diesen Prozentsatz durch 5,83, d.h. die vollständige Diagrammhöhe in Zoll. Das Ergebnis wird dann von 6,25 abgezogen, also vom unteren Bereich der Grafik. So erhalten wir die höchste Position für das Diagramm auf der Seite. Die Endposition 'Y' lautet immer 6,25, weil dies die unterste Diagrammzeile ist. Nachdem die einzelnen Balken ausgegeben wurden, wird der Text für den Balken gemeinsam mit dem Wert für den numerischen Balken selbst ausgegeben. Der Text für jeden Balken wird als gedrehter Text unterhalb der untersten Diagrammzeile ausgegeben. Der numerische Wert wird direkt oberhalb der Oberseite der Leiste selbst ausgegeben.

Erstellen Sie das Programm mit Hilfe des Befehls Create Bound RPG Program (CRTBNDRPG):

```
CRTBNDRPG PGM(lib/BGR001RG) SRCFILE(lib/QRPGLESRC) SRCMBR(BGR001RG)
```

Nachdem Sie die Druckerdatei und das Programm kompiliert haben, können Sie ein Musterdiagramm erstellen, indem Sie den folgenden Befehl CALL ausführen:

```
CALL BGR001RG PARM('SAMPLE GRAPH' 0 100000 12
'030000025000050000065000075000095000045000015000022000032000098500008500'
'January  February  March  April  May  June  July
August  September  October  November  December')
```

Der fünfte und sechste Parameter sind eigentlich Arrays, die die Diagrammwerte und Überschriften enthalten. Die Ausgabe dieses Diagramms sollte der Ausgabe in Abbildung 4.17 ähneln.

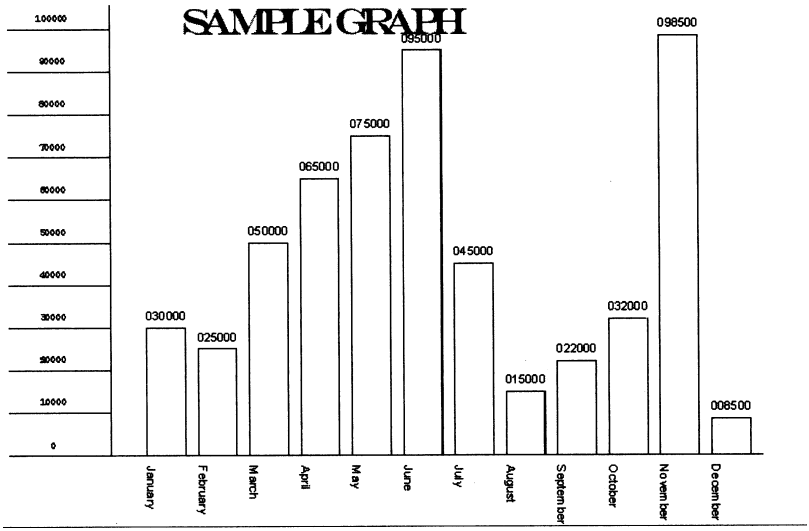


Abbildung 4.17: Dieses Balkendiagramm wird mit den AFP DDS-Schlüsselwörtern erstellt.

Wie Sie sehen, können mit sehr wenig Code beeindruckende Ergebnisse erzielt werden. Möglich wird das mit dem Advanced Function Printing. Sie wissen nun, wie einfach es ist, mit dem Schlüsselwort OVERLAY ein Vordruckformular mit einem Laserdrucker-Ausdruck eines Formulars zu ersetzen. Wir haben außerdem gemeinsam ein flexibles Programm für das Zeichnen von Balkendiagrammen erstellt, das problemlos in andere Anwendungen integriert werden kann. Das alles, ohne dass Sie zusätzliche Software kaufen müssen. Dazu verhilft Ihnen die große Leistungsfähigkeit von AFP.

Teil II
Cooles zum PC-Client

Die Integration von PCs mit der iSeries und der AS/400 ist in den letzten Jahren exponentiell gestiegen. Das liegt meist an zwei Faktoren: Erstens wurde im Laufe der Entwicklung von PC Support zu Client Access die Funktionalität für die gemeinsame Nutzung von Informationen und Ressourcen mit PC und der 400 sowohl einfacher als auch nahtloser. Die Möglichkeit, mit Open Database Connectivity (ODBC) auf DB2/400-Daten von einer PC-Anwendung aus – wie z.B. Microsoft Access – zuzugreifen, eröffnet eine neue Welt von Entwicklungsoptionen. Vor kurzem wurde außerdem die DB2/400-Datenbank mit einem OLE-DB-Treiber erweitert und damit die Erstellung von Client-Anwendungen flexibilisiert. Über den Einsatz von ActiveX Data Objects (ADO) können Sie problemlos Programme in Entwicklungsumgebungen – wie z.B. Visual Basic oder Microsoft Access – erstellen, von denen Zugriff auf Datensatzebene erfolgt, und Sie können Ihre DB2/400-Datenbank aktualisieren. Der zweite Faktor, der für eine engere PC-zu-400-Integration förderlich ist, ist TCP/IP – oder genauer gesagt: FTP. Der FTP-Server von OS/400 ermöglicht Ihnen nicht nur die Übertragung von Dateien. Sie können OS/400-Befehle in einer FTP-Sitzung ausführen, wenn Sie am FTP-Server von OS/400 angemeldet sind. Das kann ein hervorragendes Tool sein, um Sie dabei zu unterstützen, eine vorhandene PC- oder LAN-Anwendung mit einer vorhandenen Anwendung auf der 400 zu integrieren.

In diesem Abschnitt beschäftigen wir uns mit der Frage, wie beide Funktionen eingesetzt werden können, um die Integration von PC-Anwendungen mit der 400 zu verbessern.

5

ADO, OLE DB und andere Akronyme

Seit mehreren Jahren lautet das Akronym in der Domäne der Datenbankkonnektivität zwischen Windows und DB2/400: ODBC. Außer dieser gut funktionierenden Option gibt es jedoch noch weitere Akronyme, die im Zusammenhang mit der Geschwindigkeit von ODBC häufig genannt werden. Vor kurzem wurden neue Optionen verfügbar, mit denen die Verbindung mit der Datenbank der 400 nicht nur schneller, sondern auch einfacher wird. *ActiveX Data Objects* (ADO) ermöglichen es Ihnen, Dateien aus einer Datenquelle einzulesen, wobei Standardbefehle der Structured Query Language (SQL) verwendet werden. Damit wird die Anwendungsentwicklung vereinfacht, während gleichzeitig die Flexibilität erhöht wird.

Obwohl ADO mit ODBC verwendet werden kann, hat diese Situation auch ihre Nachteile. Wenn Sie z.B. eine Verbindung mit einer Tabelle in einer Bibliothek aufnehmen möchten, die nicht im ODBC-Treiber aufgeführt wird, müssen Sie den Treiber modifizieren. Außerdem wird damit Ihre Anwendung abhängig von der spezifischen ODBC-Datenquelle. Dies kann ein Problem werden, wenn Sie Programme für die Verwendung durch mehrere Benutzer entwickeln, weil Datenquellen für Benutzer erstellt werden müssen, die über die Datenquelle nicht verfügen. Mit der Unterstützung für *Object Linking Environment Data Base* (OLE DB) ist dieses Problem gelöst. Der Grund dafür ist, dass Ihre Anwendungen, sobald Sie den OLE DB-Treiber von IBM, der mit Client Access ausgeliefert wird, installiert haben, auf die Daten nicht mehr mit dem ODBC-Datenquellennamen, sondern mit dem Systemnamen der DB2/400 zugreifen, mit dem Sie eine Verbindung aufnehmen.

Die Anforderungen für die Verwendung von OLE DB sind minimal. Die DB2/400 OLE DB-Unterstützung ist Teil von Client Access Express. Unterstützung von ActiveX Data Object (ADO) ist Teil des Windows-Betriebssystems. So lange Sie also über Windows und Client Access verfügen, sind Sie startklar. Für OLE DB-Unterstützung sind zwar einige Fremdanbieteroptionen vorhanden, in unserem Beispiel jedoch verwenden wir den OLE DB-Treiber von IBM.

Als Erstes möchten wir erklären, wie ActiveX-Datenobjekte funktionieren. Anwendungen, die ADO verwenden, bestehen aus drei Hauptkomponenten:

1. Verbindungsobjekt
2. Befehlsobjekt
3. Recordset-Objekt

Das Verbindungsobjekt wird verwendet, um die Verbindungsinformationen für unsere Daten zu definieren. Dazu gehören der Benutzername, das Passwort, das verwendet werden soll, um Zugriff auf die Daten zu erhalten, der Systemname für die Verbindung sowie der Anbietername – d.h., im Fall des OLE DB-Treibers: IBMDA400. Das Befehlsobjekt legt fest, was mit der definierten Verbindung geschehen soll. Das geschieht normalerweise mit einer Structured-Query-Language- (SQL-) Anweisung. Diese kann auch verwendet werden, um Befehle oder ein Programm aufzurufen. Das Recordset-Objekt wird für den Zugriff auf die eigentlichen Daten verwendet, die im Befehlsobjekt definiert sind. Die ADO-Objekt-Hierarchie wird in Abbildung 5.1 gezeigt.

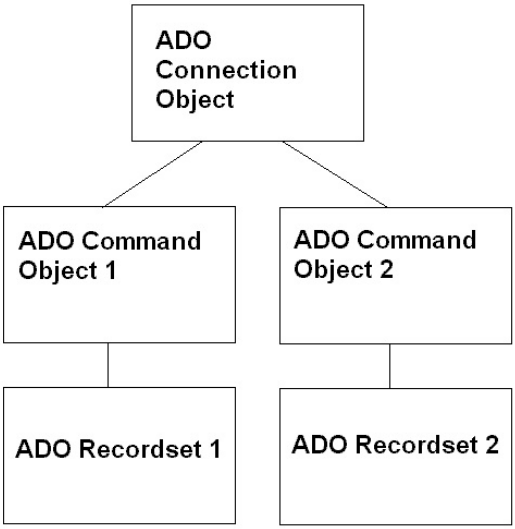


Abbildung 5.1: Die ADO-Objekthierarchie

Wie Sie sehen können, sind mehrere Befehls- oder Recordset-Objekte pro Verbindung vorhanden. Der Grund dafür ist, dass unser Verbindungsobjekt maschinenspezifisch ist. Wir benötigen nur eine Verbindung pro Maschine, mit der wir eine Verbindung herstellen. Sie werden auch bemerken, dass ein Recordset direkt mit einer Verbindung bzw. über ein Befehlsobjekt verknüpft werden kann. Mehrere Befehls-/Recordsets können verwendet werden, wenn Sie gleichzeitig auf mehrere Dateien zugreifen müssen. Ein Beispiel hierfür wäre, wenn Sie einen Datensatz aus einer Datei einlesen und – basierend auf dem Wert eines bestimmten Felds – einen Wert von einer anderen Datei einlesen müssen. Wenn mehrere Dateien an verschiedenen Stellen des Programms eingelesen werden müssen, kann ein Befehlsobjekt neu definiert und wiederverwendet werden.

File Viewer-Anwendung

Wir verwenden in unserem Beispiel Visual Basic 6.0. Vor der Verwendung von ADO ab VB 6.0 müssen Sie in der Microsoft ActiveX Data Objects Library eine Referenz hinzufügen, indem Sie in das Projektmenü wechseln und „References“ auswählen. Der Dialog „Reference“, der in Abbildung 5.2 gezeigt wird, erscheint. Blättern Sie in der Liste, bis Sie die neueste Version der gezeigten Microsoft ActiveX Data Objects Library sehen. In Ihrem System befinden sich eventuell mehrere Versionen: Wenn Sie die neueste Version verwenden, können Sie die Kompatibilität gewährleisten. Nun sind Sie bereit, um mit ADO-Objekten zu arbeiten.

Die Verbindung mit Ihrem System wird mit Hilfe der Eigenschaft ConnectionsString des Objekts Connection definiert. Dies geschieht in Visual Basic 6.0 mit Hilfe des folgenden Codes:

```
DIM Conn AS ADODB.Connection

Conn.ConnectionString= "Provider=IBMDA400;User ID=USER;
Password=PASSWORD;Data Source=SYSNAME"
```

In diesem Beispiel definieren wir als Erstes das Verbindungsobjekt Conn als Objekt ADODB.Connection, indem wir Befehl DIM eingeben. Die Eigenschaft ConnectionString wird als Nächstes definiert. Wie ich bereits erwähnte, lautet der Provider-Wert IBMDA400 für den DB2/400 OLE DB-Provider. Die Benutzer-ID und die Passwörter können weggelassen werden; damit sorgen Sie dafür, dass ein Dialog erscheint, der den Benutzer auffordert, einen Benutzernamen und ein Passwort einzugeben. Der Wert für Data Source schließlich sollte Ihrem Systemnamen entsprechen –

wie innerhalb von Client Access definiert. Diesen Namen finden Sie, indem Sie den Operations Navigator starten und den Eintrag unter „My AS/400 Connections“ prüfen.

Ein Recordset-Objekt wird verwendet, um auf die eigentlichen Daten zuzugreifen, die Sie aus Ihrer Datenquelle abgerufen haben. Die Daten für den Abruf über ADO werden mit Hilfe eines Standard-SQL-Strings definiert. Zugriff auf die Felder selbst erfolgt über die Sammlung „Fields“ im Recordset-Objekt. Abbildung 5.3 zeigt ein Beispiel dafür, wie Sie ein Verbindungs- und Recordset-Objekt einsetzen, um Daten mit Hilfe des DB2/400 OLE DB-Providers auszulesen.

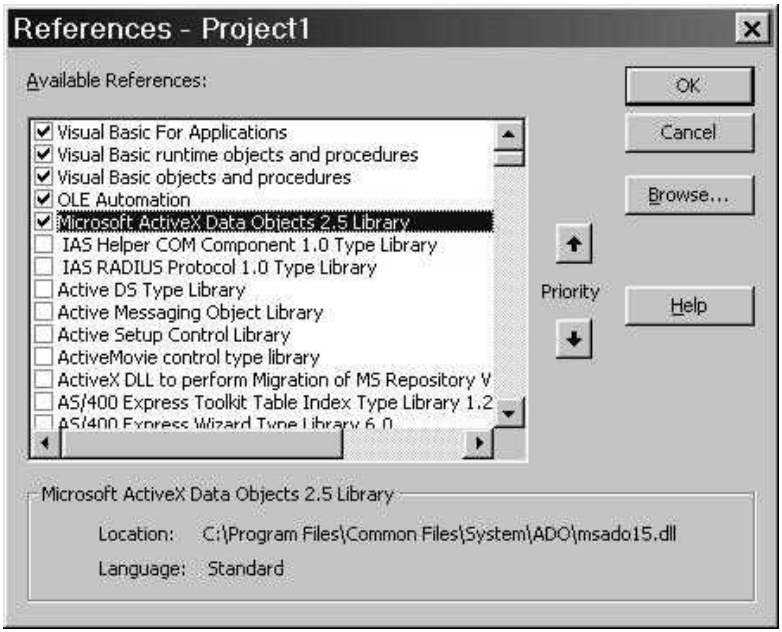


Abbildung 5.2: Auswahl der Microsoft ActiveX Data Objects Library

```
Dim rs As New ADODB.Recordset, ConnectionString As String
Dim conn As New ADODB.Connection, cmd As New ADODB.Command, RecSrc As
String
conn.Provider = "IBMDA400"
conn.Properties("Data Source") = "S1234ABC"
conn.Open
Set rs.ActiveConnection = conn
rs.Open "SELECT * FROM QSYS2.SYSTABLES WHERE TABLE_SCHEMA='QSYS'"
rs.MoveFirst
Do Until rs.EOF
Debug.Print rs.Fields("TABLE_NAME"), rs.Fields("TABLE_TEXT")
rs.MoveNext
Loop
```

Abbildung 5.3: Ein Beispiel für die Verwendung von ActiveX-Datenobjekten in VB 6

In diesem Beispiel definieren wir das Verbindungsobjekt mit Hilfe der Provider- und Data-Source-Eigenschaften anstelle mit der Eigenschaft ConnectionString. Mit beiden Methoden wird das gleiche Ergebnis erzielt. Dieser Beispielcode zeigt eine Liste von Dateien und Beschreibungen im Visual-Basic-Debug-Fenster an. Die Datei wird mit folgender SQL-Anweisung definiert:

```
SELECT * FROM QSYS2.SYSTABLES WHERE TABLE_SCHEMA='QSYS'
```

Die Datei SYSTABLES in QSYS2 führt eine Liste aller Dateien in Ihrem System gemeinsam mit wichtigen Informationen über die einzelnen Tabellen. Die meisten Beispiele in diesem Kapitel verwenden diese Tabelle gemeinsam mit Tabelle SYSCOLUMNS, die für die einzelnen Dateien in Datei SYSTABLES Feldinformationen aufweist. Dies geschieht, weil es sich um Dateien handelt, die im gesamten System verfügbar sind. Zusätzlich ermöglicht uns dies, einige nützliche Beispielprogramme zu erstellen, indem wir die bereitgestellten Informationen in diesen Dateien nutzen.

Das ADO-Befehlsobjekt hat zwei Aufgaben. Die erste Aufgabe besteht darin, die Datei zu definieren, die von Ihrem Recordset-Objekt verwendet werden soll. Die zweite Aufgabe ist die Ausführung eines OS/400-Programms oder -Befehls. Sie werden sich vielleicht fragen, warum Sie die Datei mit einem Befehlsobjekt definieren sollen, wenn Sie die Datei auch mit dem Recordset-Objekt definieren können. Ein wichtiger Grund dafür ist, dass das ADO-Befehlsobjekt Parameter zulässt. Beim Zugriff auf eine Datei unter Verwendung eines SQL-Anweisungsparameters können Sie in die Anweisung Werte einfügen, ohne die gesamte SQL-Anweisung neu zu erstellen. Damit wird die Leistung im Vergleich zur Neuerstellung der gesamten SQL-Anweisung verbessert. Wir können z.B. den folgenden VB-Code in unsere SQL-Anweisung integrieren:

```
Dim Cmd As New ADODB.Command, Rs As New ADODB.Recordset,
      Conn As New ADODB.Connection

Conn.Provider = "IBMDA400"
Conn.Properties("Data Source") = "S1234ABC"
Conn.Open

Set Cmd.ActiveConnection = Conn

Library = "QSYS"

Cmd.CommandText = "SELECT * FROM QSYS2.SYSTABLES WHERE
      TABLE_SCHEMA='" & _
                  Library & "'"

Set Rs = Cmd.Execute

Rs.MoveFirst
```

Das Resultat ist, dass wir jedes Mal, wenn wir die Library-Variable ändern möchten, die gesamte Anweisung neu erstellen müssen. Andererseits können wir diesen Code mit dem folgenden Code neu erstellen, indem wir Parameter verwenden:

```
Dim Cmd As New ADODB.Command, Rs As New ADODB.Recordset,
      Conn As New ADODB.Connection, ConnStr As String

Conn.Provider = "IBMDA400"
Conn.Properties("Data Source") = "S1234ABC"
Conn.Open

With Cmd
      .ActiveConnection = Conn
      .CommandText = "SELECT * FROM QSYS2.SYSTABLES WHERE
TABLE_SCHEMA=?"
       .Parameters(0) = "QSYS"
End With

Set Rs = Cmd.Execute

Rs.MoveFirst
```

Obwohl mit beiden Techniken das gleiche Ergebnis erzielt werden kann, wird die Leistung verbessert, wenn Parameter zum Einsatz kommen, weil bei jeder Neuerstellung der SQL-Anweisung diese über den Query Optimizer gesendet werden muss. Wenn Sie Werte mit Hilfe von Parametern einfügen, ist das nicht mehr erforderlich.

Der Vorteil der Verwendung von ADO ist es, dass Sie den Code wieder verwenden können, weil die gleichen Schritte für die Verbindung mit allen Dateien erforderlich sind. Sie können diese Technik auch verwenden, um Daten aus Visual Basic, VBA oder sogar von einer Active Server Page einzulesen, wobei VB Scripting verwendet wird.

Dabei gehen Sie wie folgt vor:

1. Definition des Verbindungsobjekts,
2. Definition der Daten für den Abruf mit Hilfe einer SQL-Anweisung oder eines Befehlsobjekts,
3. Öffnen eines Recordsets mit Hilfe der vorbereiteten SQL-Anweisung oder des Befehlsobjekts,
4. Einlesen des Recordsets,
5. Verarbeitung der Daten,
6. Schließen des Recordsets,
7. Schließen des Befehlsobjekts, falls erforderlich,
8. Schließen der Verbindung.

Dieser Prozess ähnelt den Schritten, die für das Einlesen von Daten aus einer Datei in einem RPG-Programm erforderlich sind. Betrachten Sie das Verbindungsobjekt wie Ihre RPG F-Spezifikationen – wir verwenden es, um zu definieren, wo unsere Daten herstammen. Mit dem Befehlsobjekt oder der SQL-Anweisung im Befehl „Recordset.Open“ wird auf die gleiche Weise definiert, welche Daten abgerufen werden sollen, wie mit einer Key List (KLIST) in einem RPG-Programm. Recordset.MoveNext schließlich funktioniert wie eine READ-Operation innerhalb eines RPG-Programms.

Unter Einsatz der neuen Funktionen erstellen wir ein einfaches Beispielprogramm mit ADO und Visual Basic 6.0. Das Programm, das wir erstellen, generiert ein Explorer-ähnliches Fenster, in dem Bibliotheken und Dateien aufgelistet werden. Es erlaubt dem Be-

nutzer, Feldinformationen anzuzeigen, indem er auf einen Dateinamen klickt. Starten Sie VB 6 und, wenn der Dialog „New Project“ angezeigt wird, wählen Sie Standard EXE. Wenn das Formular für Ihr neues Projekt angezeigt wird, wechseln Sie in den Bildschirm „Properties“ und ändern die Parameter auf die Werte, die Sie in Tabelle 5.1 sehen.

Tabelle 5.1: Parameter und Werte für unser Formularobjekt

Parameter	**Wert**
Name	frmFileViewer
Border Style	3 – Fixed Dialog
Caption	File Viewer
Height	5365
Width	8280

Nun wechseln Sie in das Menü „Project“ und wählen „References“. Wählen Sie Microsoft ActiveX Data Objects Library wie bereits beschrieben wurde.

Wir müssen diesen Schritt ausführen, um die ADO-Funktionalität unserer Anwendung zur Verfügung zu stellen. Nun müssen wir sicherstellen, dass die anderen Objekte, die wir in unserem Formular verwenden, verfügbar sind. Dies geschieht, indem Sie mit der rechten Maustaste auf das Toolbox-Fenster klicken und „Components“ wählen. Wählen Sie in dem Dialogfeld, das in Abbildung 5.4 gezeigt wird, Microsoft Windows Flex Grid Control und Microsoft Windows Common Controls. Das Flex Grid Control wird verwendet, um unsere Feldinformationen anzuzeigen. Im Fenster „Windows

Common Controls“ verwenden wir das Tree-View-Objekt für die Anzeige der Bibliotheks- und Dateinamen in einem „Explorer“-Format.

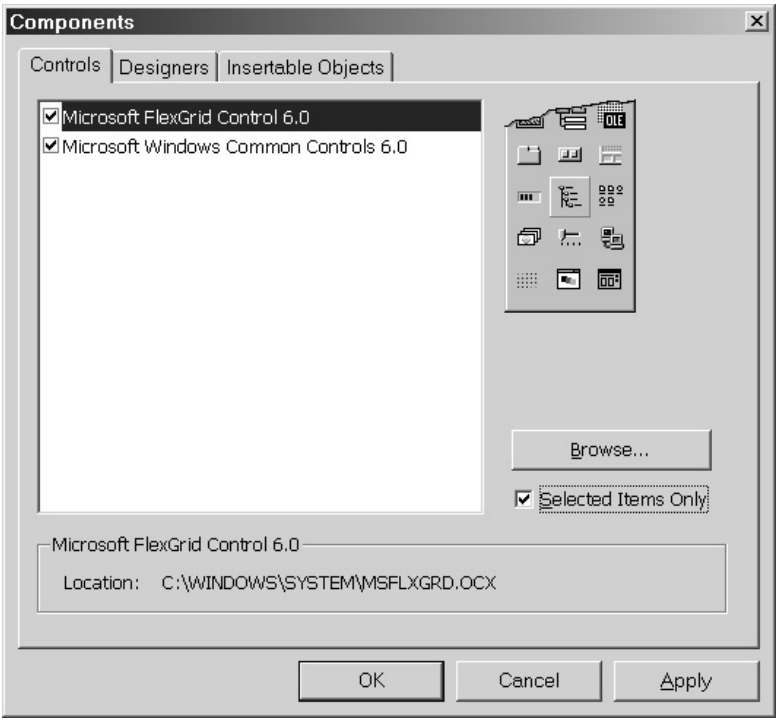


Abbildung 5.4: Visual-Basic-Komponenten werden von diesem Dialog hinzugefügt.

Nun müssen wir unser Formular mit einigen dieser Objekte ergänzen. Als Erstes wählen wir das Symbol Tree View im Toolbox-Fenster aus. Dieses Symbol erinnert an eine Liste vom Explorer-Typ. Sie können dieses Symbol auch identifizieren, indem Sie den Mauszeiger über dem Symbol positionieren, bis das Wort TreeView

angezeigt wird. Nun verschieben Sie den Mauszeiger zu Ihrem Formular und ziehen den Zeiger über einen kleinen Bereich Ihres Formulars. Wechseln Sie in das Fenster „Properties“ für dieses Objekt und setzen Sie die einzelnen Parameter auf den entsprechenden Wert, der in Tabelle 5.2 gezeigt wird.

Sobald Sie diese Werte eingegeben haben, sollte das Tree-View-Objekt die obere Hälfte Ihres Formulars belegen. Nun verwenden wir den gleichen Prozess, um ein Flex-Grid-Objekt zu erstellen. Dies wird verwendet, um die Feldinformationen für die ausgewählte Datei anzuzeigen. Wählen Sie das Symbol Flex Grid aus dem Toolbox-Fenster und erstellen Sie ein neues Flex Grid, indem Sie einen kleinen Bereich des nicht verwendeten Bereichs des Formulars ziehen. Nun verwenden Sie das Fenster „Properties“, um die in Tabelle 5.3 gezeigten Werte für die Parameter einzustellen.

Tabelle 5.2: Parameter und Werte für unser Tree-View-Objekt

Parameter	**Wert**
Name	tvwLibFile
Height	2655
Indentation	300
Left	120
LineStyle	1
Top	120
Width	7935

Tabelle 5.3: Parameter und Werte für unser Flex-Grid-Objekt	
Parameter	**Wert**
Name	mfgFieldInfo
Cols	5
FixedCols	0
FixedRows	1
Height	2295
Left	120
Top	2760
Width	7935

Wenn der Parameter Cols auf 5 gesetzt wird, können wir den Anzeigefeldnamen, Text, Typ, Breite und Dezimalstellen anzeigen. Die Parameter FixedCols und FixedRows legen die erste Zeile für Spaltenüberschriften fest. Die Parameter für Höhe und Breite (height/width)werden verwendet, um die Größe unseres Objekts zu ändern. Die Parameter für links und oben (left/top) bestimmen die Position des Objekts abhängig von seiner oberen linken Ecke. Sobald Sie diese Parameter festgelegt haben, sollte Ihr Formular wie in Abbildung 5.5 gezeigt aussehen.

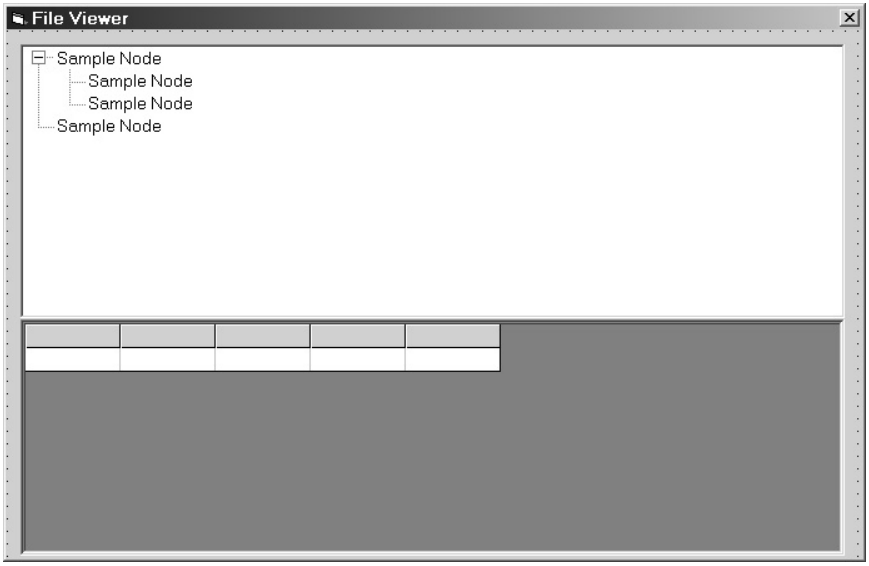


Abbildung 5.5: Das Visual Basic 6.0 ADO File Viewer Beispiel-Anwendungsformular.

Nachdem wir nun das Formular erstellt haben, müssen wir den Code hinzufügen. Als Erstes fügen wir Deklarationen für unsere ADODB-Objekte in den Abschnitt General Declarations unseres Formularmoduls ein. Um die Codeansicht anzuzeigen, wählen Sie im Menü „View“ die Option „Code“. Nun wählen Sie im Objektlistenfeld auf der linken Seite des Codefensters die Option „General“ und wählen aus dem Prozedurlistenfeld rechts die Option „Declarations“. Geben Sie in diesen Bereich folgenden Code ein:

```
Dim Rs As New ADODB.Recordset, Conn As New ADODB.Connection, Cmd As
        New ADODB.Command
```

Damit erhalten wir ein Recordset-Objekt mit der Bezeichnung RS, ein Verbindungsobjekt mit der Bezeichnung Conn und ein Befehlsobjekt namens Cmd. Nun müssen wir den Code erstellen, von dem die Bibliothek und die Dateiinformationen aus SYSTABLES in unsere Baumansicht geladen werden. Fügen Sie den Code in Abbildung 5.6 in das Load-Event Ihres Formularobjekts ein:

```
Private Sub Form_Load()

Conn.ConnectionString = "Provider=IBMDA400;Data Source=S1234ABC;"
Conn.Open

Rs.ActiveConnection = Conn
Rs.Open "SELECT TABLE_SCHEMA, TABLE_NAME, TABLE_TEXT FROM
QSYS2.SYSTABLES WHERE TABLE_TYPE='P' AND FILE_TYPE='D' " & _
                  "ORDER BY TABLE_SCHEMA, TABLE_NAME"

Rs.MoveFirst

Do Until Rs.EOF
If LIB Rs.Fields(“TABLE_SCHEMA”) Then
TvwLibFile.Nodes.Add , 2, Rs.Fields("TABLE_SCHEMA"),
Rs.Fields("TABLE_SCHEMA")
LIB = Rs.Fields("TABLE_SCHEMA")
End If
X = X + 1
TvwLibFile.Nodes.Add LIB, 4, Str$(X) & "_" & Rs.Fields("TABLE_NAME"),
Rs.Fields("TABLE_NAME") & " " & Rs.Fields("TABLE_TEXT")
Rs.MoveNext
Loop
Rs.Close

End Sub
```

Abbildung 5.6: Dieses Formular fügt Bibliothek und Dateiinformationen in ein Tree-View-Objekt ein.

Mit dieser Prozedur wird zuerst das Verbindungsobjekt definiert, danach das Recordset-Objekt mit Hilfe einer SQL SELECT-An-

weisung, von der der Bibliotheksname, der Dateiname und der Dateitext für physische Datendateien ausgewählt wird. Die Prozedur liest anschließend Ihren Recordset ein und fügt bei jeder Änderung im Bibliotheksnamen (TABLE_SCHEMA) einen Eintrag auf Root-Ebene in unsere Baumansicht ein. Für jeden eingelesenen Datensatz werden Dateinamen und Dateitext konkateniert und als Kind des Bibliotheksnamenseintrags auf Root-Ebene hinzugefügt. Wenn die Anwendung ausgeführt wird, werden nur Bibliotheksnamen angezeigt. Wenn Sie die Bibliotheksansicht erweitern, wird die Dateiliste angezeigt.

Als Nächstes müssen wir den Code erstellen, um die Feldinformationen für eine ausgewählte Datei auszuwählen und anzuzeigen. Verwenden Sie den in Abbildung 5.7 gezeigten Code als Doppelklick-Ereignis (DblClick) für das Objekt tvwLibFile.

```
Private Sub tvwLibFile_DblClick()

On Error Resume Next
Blnk = InStr(1, tvwLibFile.SelectedItem, " ")
If Blnk = 0 Then Blnk = Len(Trim(tvwLibFile.SelectedItem))
FName = Left(tvwLibFile.SelectedItem, Blnk)

If tvwLibFile.SelectedItem.Children = 0 Then
LIB = tvwLibFile.SelectedItem.Parent

If FName LIB And FName  "" Then
If Cmd.CommandText = "" Then
Cmd.ActiveConnection = Conn
Cmd.CommandText = "SELECT * FROM QSYS2.SYSCOLUMNS WHERE
SYSTEM_TABLE_NAME=? "&_
                  "AND TABLE_SCHEMA=?   ORDER BY ORDINAL_POSITION"
End If
Cmd.Parameters(0) = Fname
Cmd.Parameters(1) = LIB
```

Abbildung 5.7: Das Doppelklick-Ereignis des Tree-View-Objekts zeigt Feldinformationen an (Teil 1 von 2).

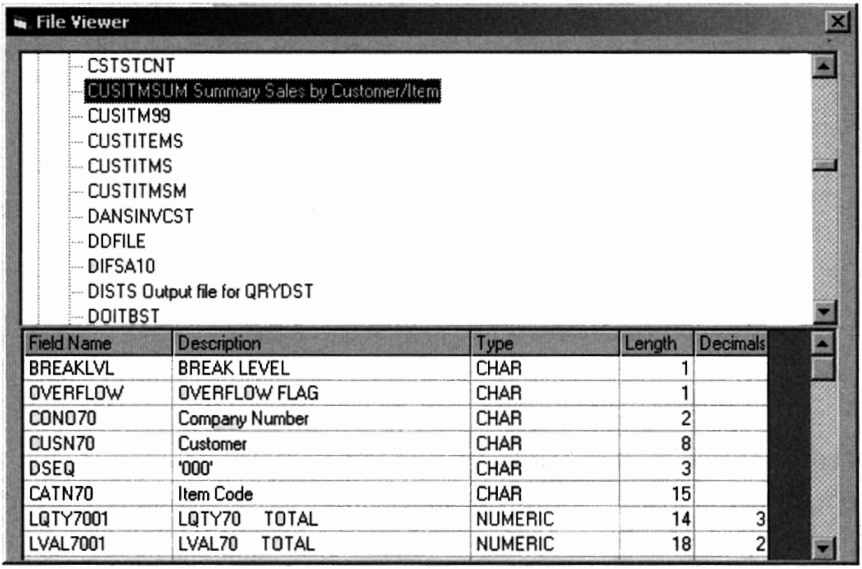
```
Set Rs = Cmd.Execute
mfgFieldInfo.Clear
mfgFieldInfo.Rows = 1
mfgFieldInfo.Row = 0
mfgFieldInfo.Col = 0
mfgFieldInfo.Text = “Field Name”
mfgFieldInfo.ColWidth(0) = 1440
mfgFieldInfo.Col = 1
mfgFieldInfo.Text = “Description”
mfgFieldInfo.ColWidth(1) = 2880
mfgFieldInfo.Col = 2
mfgFieldInfo.Text = “Type”
mfgFieldInfo.ColWidth(2) = 1440
mfgFieldInfo.Col = 3
mfgFieldInfo.Text = “Length”
mfgFieldInfo.ColWidth(3) = 720
mfgFieldInfo.Col = 4
mfgFieldInfo.Text = “Decimals”
mfgFieldInfo.ColWidth(4) = 720
mfgFieldInfo.Rows = mfgFieldInfo.Rows + 1
Rs.MoveFirst
Do Until Rs.EOF
mfgFieldInfo.Col = 0
mfgFieldInfo.Text = Rs.Fields(“SYSTEM_COLUMN_NAME”)
mfgFieldInfo.Col = 1
mfgFieldInfo.Text = IIf(IsNull(Rs.Fields(„COLUMN_TEXT“)), „“,
Rs.Fields(“COLUMN_TEXT”))
mfgFieldInfo.Col = 2
mfgFieldInfo.Text = Rs.Fields(“DATA_TYPE”)
mfgFieldInfo.Col = 3
mfgFieldInfo.Text = Rs.Fields(“LENGTH”)
mfgFieldInfo.Col = 4
mfgFieldInfo.Text = IIf(IsNull(Rs.Fields(“NUMERIC_SCALE”)), “ “,
Rs.Fields(“NUMERIC_SCALE”))
mfgFieldInfo.Rows = mfgFieldInfo.Rows + 1
mfgFieldInfo.Row = mfgFieldInfo.Row + 1

Rs.MoveNext
Loop
Rs.Close

End If
End If

End Sub
```

Abbildung 5.7: Das Doppelklick-Ereignis des Tree-View-Objekts zeigt Feldinformationen an (Teil 2 von 2).

Diese Prozedur beginnt mit der Extrahierung des Dateinamens aus dem konkatenierten Dateinamen/Dateitext, der im Tabellen-View-Node angezeigt wird. Als Nächstes bestimmen wir den Bibliotheksnamen unter Verwendung der Eigenschaft „Parent“ der Sammlung „SelectedItem“ für das Tabellen-View-Objekt. Die Sammlung SelectedItem wird verwendet, um Informationen über ein aktuell ausgewähltes Objekt in einer Baumansicht abzurufen. Der Bibliotheksnamenswert wird auch verwendet, um festzustellen, ob das ausgewählte Objekt tatsächlich ein Bibliotheksname ist. Als Nächstes definieren wir den Befehlstext für unser Befehlsobjekt. Dies geschieht nur bei der ersten Ausführung der Anwendung. Das ist möglich, weil wir ein Befehlsobjekt verwenden, das einen Parameter für den Bibliotheksnamen und einen Parameter für den Dateinamen enthält. So können wir zusätzliche Leistungsverbesserungen implementieren. Die SQL-Anweisung wird verwendet, um unsere Feldinformationen aus der Datei SYSCOLUMNS abzurufen. Das Befehlsobjekt wird anschließend mit unserem Recordset verknüpft, indem folgender Befehl eingegeben wird:

```
Set RS = Cmd.Execute
```

Als Nächstes definieren wir die Spaltenüberschriften für unser Flex-Grid-Objekt unter Verwendung der Methoden .Row, .Col und .Text. Diese Methoden werden anschließend verwendet, um die Feldinformationen aus unserem Recordset in das Flex Grid einzugeben. Nachdem Sie den gesamten Code für dieses Beispiel eingegeben haben, drücken Sie die Taste F5, um die Anwendung auszuführen. Sie können eine Bibliotheksansicht erweitern, indem Sie auf das Pluszeichen neben der Bibliothek klicken. Danach doppelklicken Sie auf einen Dateinamen, um die Feldinformationen für diese Datei anzuzeigen. Abbildung 5.8 zeigt, wie das fertige FileViewer-Anwendungsfenster aussieht.



Field Name	Description	Type	Length	Decimals
BREAKLVL	BREAK LEVEL	CHAR	1	
OVERFLOW	OVERFLOW FLAG	CHAR	1	
CONO70	Company Number	CHAR	2	
CUSN70	Customer	CHAR	8	
DSEQ	'000'	CHAR	3	
CATN70	Item Code	CHAR	15	
LQTY7001	LQTY70 TOTAL	NUMERIC	14	3
LVAL7001	LVAL70 TOTAL	NUMERIC	18	2

Abbildung 5.8: Die fertigen FileViewer-Anwendungen werden verwendet, um Datei/Feld-Informationen anzuzeigen.

Berichtsschreibung

Wie bereits diskutiert, gehört es zu den Vorteilen von ADO und OLE DB, dass Sie Anwendungen mit Hilfe von beliebigen Entwicklungstools mit einer großen Auswahl entwickeln können. Wir haben uns bisher mit dem Thema befasst, wie eine Anwendung mit Hilfe von Visual Basic 6.0 erstellt wird. Nun beschäftigen wir uns mit der Frage, wie mit Microsoft Access 200 via ADO und DB2/400 OLE DB-Treiber ein Bericht erstellt werden kann. Dieser Prozess ähnelt dem Prozess, der verwendet, wird, um die Feldinformationen in unser Flex Grid zu laden. Der Unterschied dabei ist, dass wir diese Daten in eine Microsoft Access-Tabelle laden und anschließend einen Access-Bericht aus diesen Daten erstellen.

Die erforderlichen Schritte für dieses Beispiel entsprechen denen, die in unserem ersten Beispiel verwendet wurden, davon ausgenommen die Tatsache, dass wir statt Visual Basic 6.0 Microsoft Access 200 verwenden. Um Access 200 zu starten, öffnen Sie das Programm und erstellen eine neue leere Access-Datenbank. Wählen Sie Registerkarte „Tabellen, Neu“ und wählen Sie im angezeigten Dialogfeld „Entwurfsansicht“. Verwenden Sie die Werte, die in Tabelle 5.4 gezeigt werden, um unsere Tabelle mit den erforderlichen Feldern zu ergänzen.

Tabelle 5.4: Felddefinitionen für die Access-Tabelle tblFieldInfo		
Feldnamen	**Typ**	**Feldlänge**
LibraryName	TEXT	10
FileName	TEXT	10
FieldName	TEXT	10
FieldText	TEXT	30
FieldType	TEXT	8
FieldLength	NUMBER	LONG INTEGER
FieldDecimals	NUMBER	LONG INTEGER

Nachdem Sie alle erforderlichen Felder erstellt haben, markieren Sie die drei ersten Felder wie in Abbildung 5.9 gezeigt und wählen das Schlüsselsymbol in der Menüleiste „Tabellenentwurf“. Nun speichern wir die neue Tabelle als tblFieldInfo.

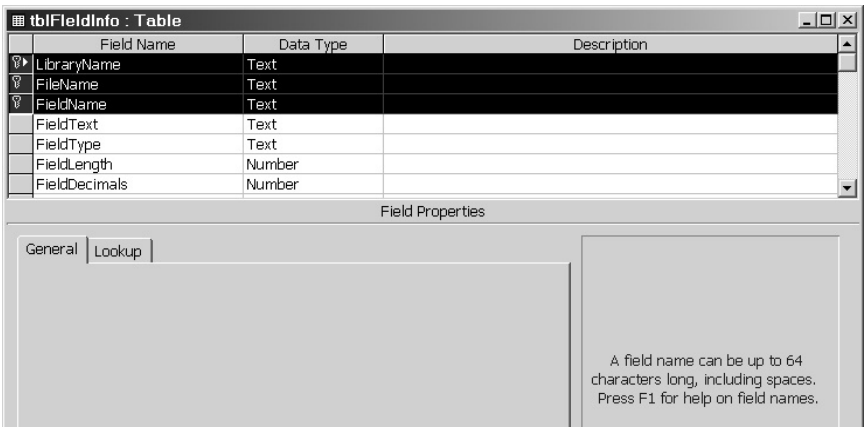


Abbildung 5.9: Verwenden Sie diesen Bildschirm, um den Dateischlüssel für die Datei tblFieldInfo festzulegen.

Diese Datei wird verwendet, um eine lokale Kopie der Felddaten aus der Datei SYSTABLES zu speichern. Wir könnten zwar alle diese Daten in die Tabelle tblFieldInfo einlesen, aber dann müssten wir mehr Datensätze verarbeiten als wenn wir einfach alle Bibliotheks- und Dateinamen einlesen und dann nur die für die Datei erforderlichen Felddaten. Damit erhält unsre Anwendung größere Effizienz. Tabelle 5.5 enthält die Felddefinitionen für die Access-Tabelle tblLibFile.

Tabelle 5.5: Felddefinitionen für die Access-Tabelle tblLibFile

Feldnamen	Typ	Feldlänge
LibraryName	TEXT	10
FileName	TEXT	10
FileText	TEXT	50

Nachdem Sie die neue Tabelle unter tblLibFile gespeichert haben, kann die Erstellung unseres Programms beginnen. Wir müssen ein neues Access-Formular erstellen, indem wir zur Registerkarte „Formulare“ wechseln und „Neu“ wählen. Wenn der Dialog „Neues Formular“ angezeigt wird, wählen Sie einfach „Entwurfsansicht“ aus.

Wenn der Formular-Designer geöffnet wird und dabei der Dialog „Eigenschaften“ nicht angezeigt wird, öffnen Sie diesen, indem Sie in das Menü „Ansicht“ wechseln und „Eigenschaften“ auswählen. Nun setzen Sie die Eigenschaften auf die Werte, die in Tabelle 5.6 gezeigt werden.

Tabelle 5.6: Parameter und Werte für unser Access-Formular		
Objekt	**Eigenschaft**	**Wert**
Detail	Height	4”
Form	Caption	File Field Report
Form	Scroll Bars	Neither
Form	Record Selectors	No
Form	Navigation Buttons	No
Form	Dividing Lines	No
Form	Auto Resize	No
Form	Border Style	Dialog
Form	Width	6”

Sobald Sie alle hier genannten Eigenschaften festgelegt haben, wechseln Sie zum Register „Ereignis“ für das Formularobjekt, wählen das „Ereignis beim Öffnen“ und klicken auf die Schaltfläche „Build“ bzw. „Aufbauen (…)“. Wählen Sie im Dialogfeld für die Programmauswahl den Code „Builder“. Wechseln Sie im Code „Builder-Fenster“ zum Menü „Tools“ und wählen Sie „References“. Stellen Sie sicher, dass die Microsoft ActiveX Data Objects Library ausgewählt ist – wie in unserem Visual-Basic-Beispiel. Sie müssen außerdem die Microsoft DAO Object Library im Dialog „References“ aktivieren. Klicken Sie dann auf „OK“, um den Dialog zu schließen. Nun fügen Sie den Code – wie in Abbildung 5.10 gezeigt – in das Form_Open-Event ein.

```
Private Sub Form_Load()
Dim Rs1 As DAO.Recordset

Conn.ConnectionString = "Provider=IBMDA400;Data Source=S1234ABC;"
Conn.Open

DoCmd.SetWarnings False
DoCmd.RunSQL "Delete * From tblLibFile;"
DoCmd.SetWarnings True

Rs.ActiveConnection = Conn
Rs.CacheSize = 512
Rs.Open "SELECT TABLE_SCHEMA, TABLE_NAME, TABLE_TEXT FROM
QSYS2.SYSTABLES WHERE TABLE_TYPE='P' AND FILE_TYPE='D' " & _
                  "ORDER BY TABLE_SCHEMA, TABLE_NAME"

Rs.MoveFirst
Set Rs1 = CurrentDb.OpenRecordset("tblLibFile")

Do Until Rs.EOF
Rs1.AddNew
    With Rs1
        .Fields("LibraryName") = Rs.Fields("TABLE_SCHEMA")
        .Fields("FileName") = Rs.Fields("TABLE_NAME")
        .Fields("FileText") = IIf(Rs.Fields("TABLE_TEXT") = "", "*
NONE", _
                Rs.Fields("TABLE_TEXT"))
        .Update
    End With

Rs.MoveNext
Loop
Rs.Close

End Sub
```

Abbildung 5.10: Dieser Code lädt temporär DB2/400-Daten in eine Microsoft Access-Tabelle.

Diese Prozedur lädt die Bibliothek und die Dateinamen aus der Datei SYSTABLES in die Access-Tabelle tblLibFile. Dieser Code ähnelt dem Code, den Sie in unserem früheren Visual-Basic-Beispiel

verwendet haben, mit der Ausnahme, dass wir hier die resultierenden Daten in eine Access-Tabelle schreiben, anstatt Einträge in einem Tree-View-Objekt zu erstellen. Was Ihnen sicher auffallen wird, ist die Tatsache, dass ich die Eigenschaft CacheSize für das Recordset-Objekt auf 512 setze. Diese Eigenschaft wird verwendet, um die Anzahl der Datensätze zu definieren, die in einer einzelnen Leseoperation abgerufen werden. Wenn jeweils nur ein Datensatz verarbeitet werden muss, z.B. in einem Programm für die Datenpflege, empfiehlt es sich CacheSize mit Wert 1 zu verwenden. In einem Programm ähnlich wie unser Beispiel, in dem große Datenmengen eingelesen werden, bietet ein höherer Wert bessere Leistung. Die beste Lösung ist es, den geeigneten Wert durch Ausprobieren zu suchen. Jeder Datensatz, der in dieser Prozedur eingelesen wird, wird in die Access-Tabelle tblLibFile mit Hilfe der Methoden Rs1.AddNew und Rs1.Update eingelesen. Sie werden feststellen, dass wir ein Microsoft Data Access Object (DAO) für unsere Access-Tabelle verwenden. Das geschieht, weil es einfacher ist, die Verbindung mit einer Tabelle innerhalb der aktuellen Access-Datenbank mit dieser Methode zu definieren, anstatt ADO zu verwenden. Nachdem Sie die Eingabe des Ereignisses Form_Load beendet haben, schließen Sie das Code-Fenster.

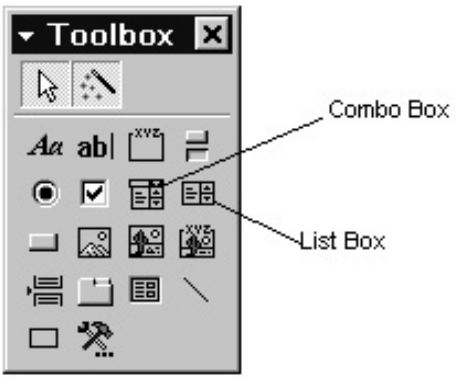


Abbildung 5.11: Verwenden Sie das Kombinationsfeld und das Listenfeld von der Access-Toolbox.

Nun müssen wir in unser Formular ein Kombinationsfeld einfügen. Dieses Feld ist für die Anzeige der Bibliotheksnamen erforderlich. Später werden wir ein Listenfeld hinzufügen, das alle Dateien innerhalb der ausgewählten Bibliothek anzeigt. Wenn Ihnen die Begriffe Kombinations- oder Listenfeld noch nicht bekannt sind, sollten Sie sich Abbildung 5.11 ansehen. Diese Grafik zeigt das Toolbox-Symbol für die einzelnen Objekte.

Wählen Sie das Symbol für das Kombinationsfeld und ziehen Sie ein neues Kombinationsfeld auf Ihr Formular. Wenn der Assistent für Kombinationsfelder angezeigt wird, klicken Sie auf „Abbrechen“. Nun klicken Sie mit der rechten Maustaste auf das Kombinationsfeld und wählen Eigenschaften. Als Nächstes verwenden Sie Tabelle 5.7, um die Eigenschaften auf die gezeigten Werte zu setzen.

Tabelle 5.7: Eigenschaften und Werte für das Kombinationsfeld

Tab	Parameter	Wert
Other	Name	CbxLibraries
Format	Left	0.7083”
Format	Top	0.2083”
Format	Width	1.8333”
Format	Height	0.1667”
Data	Row Source Type	Table/Query
Data	Row Source	SELECT DISTINCT [LibraryName] FROM tblLibFile;

Nachdem Sie die einzelnen Eigenschaften konfiguriert haben, wechseln Sie zum Menü „Ansicht“ und wählen „Code“. Wenn das Code-Fenster angezeigt wird, wählen Sie aus der Objektliste cbxLibraries und wählen in der Prozedurliste „Change“. Als Nächstes fügen Sie den Code in Abbildung 5.12 als cbxLibraries_Change Event ein. Dieses Ereignis wird jedes Mal ausgeführt werden, wenn der Eintrag in unserem Kombinationsfeld geändert wird.

```
Private Sub cbxLibraries_Change()
lbxFiles.RowSource = "SELECT FileName, FileText FROM tblLibFile
WHERE" & _
                     "LibraryName = '" & cbxLibraries.Value & "'"
lbxFiles.Requery
End Sub
```

Abbildung 5.12: Dieser Code erstellt eine Liste aller physischen Datendateien in der ausgewählten Bibliothek.

Nun fügen wir das Listenfeld hinzu, das für die Aufnahme der Dateiliste vorgesehen ist. Wählen Sie das Symbol „Listenfeld“ wie zuvor beschrieben aus der Toolbox. Nun ziehen Sie ein neues Listenfeld auf das Formular. Wenn der Assistent für Listenfelder angezeigt wird, klicken Sie auf „Cancel“. Als Nächstes klicken Sie mit der rechten Maustaste auf das Listenfeld und setzen die Eigenschaften für die in Tabelle 5.8 gezeigten Werte.

Tabelle 5.8: Eigenschaften und Werte für das Kombinationsfeld		
Tab	**Parameter**	**Wert**
Other	Name	LlbxFiles
Format	Column Count	2
Format	Column Heads	Yes
Format	Left	0.7083”
Format	Top	0.5417”
Format	Width	5.125”

Nachdem Sie diese Eigenschaften konfiguriert haben, klicken Sie auf das Menü „View“ und wählen „Code“. Nun wählen Sie vom Code-Fenster aus lbxFiles aus dem Objekt-Listenfeld und wählen DblClick aus dem Prozedur-Listenfeld. Verwenden Sie den Code in Abbildung 5.13 als Double-Click-Event für Ihr Listenfeld.

```
Private Sub lbxFiles_DblClick(Cancel As Integer)
Dim Rs1 As DAO.Recordset

DoCmd.SetWarnings False
DoCmd.RunSQL "Delete * From tblFieldInfo;"
DoCmd.SetWarnings True
If Conn.State < 1 Then
Conn.ConnectionString = "Provider=IBMDA400;Data Source=S1234ABC;"
Conn.Open
End If

If Cmd.CommandText = "" Then
Cmd.CommandText = "SELECT * FROM QSYS2.SYSCOLUMNS WHERE TABLE_NAME=?
AND TABLE_SCHEMA=? " & _
                    "ORDER BY TABLE_SCHEMA, TABLE_NAME,
ORDINAL_POSITION"
Cmd.ActiveConnection = Conn
End If
```

Abbildung 5.13: Das lbxFiles_DblClick-Ereignis lädt Felddaten in eine Access-Tabelle (Teil 1 von 2).

```
Cmd.Parameters(0).Value = lbxFiles.Value
Cmd.Parameters(1).Value = cbxLibraries.Value

Set Rs = Cmd.Execute

Rs.MoveFirst

Set Rs1 = CurrentDb.OpenRecordset("tblFieldInfo")

Do Until Rs.EOF
Rs1.AddNew
    With Rs1
        .Fields("LibraryName") = Rs.Fields("TABLE_SCHEMA")
        .Fields("FileName") = Rs.Fields("TABLE_NAME")
        .Fields("FieldName") = Rs.Fields("COLUMN_NAME")
        .Fields("FieldText") = Rs.Fields("COLUMN_TEXT")
        .Fields("FieldType") = Rs.Fields("DATA_TYPE")
        .Fields("FieldLength") = Rs.Fields("LENGTH")
        .Fields("FieldDecimals") = Rs.Fields("NUMERIC_SCALE")
        .Update
    End With
Rs.MoveNext
Loop
Rs.Close
DoCmd.OpenReport "rptFieldInfo", acViewPreview

End Sub
```

Abbildung 5.13: Das lbxFiles_DblClick-Ereignis lädt Felddaten in eine Access-Tabelle (Teil 2 von 2).

Diese Prozedur liest Feldinformationen aus SYSCOLUMNS ein und schreibt diese Daten in die Access-Tabelle tblFieldInfo, wobei die gleiche Technik zum Einsatz kommt wie zuvor, als wir diese Informationen im Visual-Basic-Beispiel in das Flex Grid einfügten. Diese Informationen werden vom Access-Bericht verwendet werden, der im nächsten Schritt unseres Beispiels erstellt wurde. Dieser Bericht wird am Ende von Abbildung 5.13 in der Print Preview-View mit dem Befehl DoCmd.OpenReport eingefügt.

Nachdem Sie den Code für das lbxFiles_DblClick eingegeben haben, speichern Sie dieses Formular als frmReportPrompt und schließen den Formular-Designer.

Nun müssen Sie den Feldinformationsbericht erstellen. Als Erstes müssen wir eine Microsoft Access-Abfrage erstellen, um unsere beiden Tabellen miteinander zu verbinden. Dies wird die Grundlage für unseren Bericht bilden.

Wählen Sie im Bildschirm der Access-Datenbank das Register „Abfragen“ und klicken Sie auf die Schaltfläche „Neu“. Wenn der Dialog „Neue Abfrage“ angezeigt wird, wählen Sie die „Entwurfsansicht“. Wenn das Dialogfeld „Tabelle anzeigen“ angezeigt wird, klicken Sie auf „Schließen“. Beide Ansichten werden in Abbildung 5.14 gezeigt.

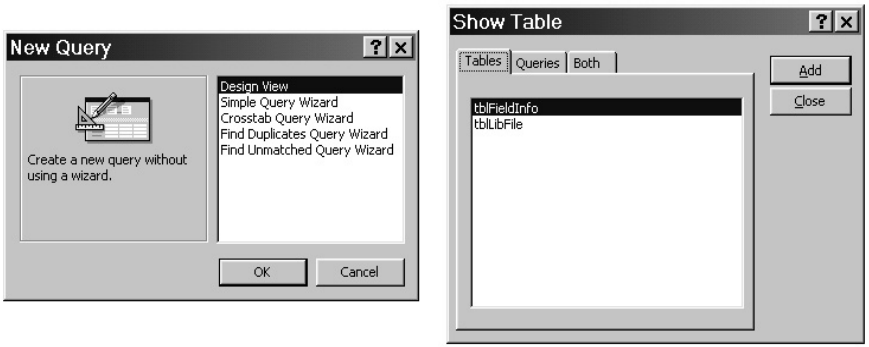


Abbildung 5.14: Dieser Dialog wird verwendet, um eine neue Microsoft Access Query zu erstellen.

Nun müssen Sie die SQL-View aus dem View-Menü auswählen. Anstatt die GUI zu verwenden, erstellen wir einfach die Abfrage, indem wir die SQL-Anweisung SELECT wie folgt eingeben:

```
SELECT [tblLibFile].[LibraryName] & "/" & [tblLibFile].[FileNAme] &
        "" & [tblLibFile].[FileText] AS FileHeader,
        tblFieldInfo.FieldName,
        tblFieldInfo.FieldText, tblFieldInfo.FieldType,
        tblFieldInfo.FieldLength, tblFieldInfo.FieldDecimals
FROM    tblLibFile
INNER JOIN tblFieldInfo ON (tblLibFile.FileName = tblFieldInfo.FileName)
AND     (tblLibFile.LibraryName = tblFieldInfo.LibraryName);
```

Nun wechseln wir zum Menü „Datei“, wählen „Speichern als“ und speichern die neue Abfrage mit dem Namen qryFieldInfo. Klicken Sie im Fenster „Datenbank“ auf das Register „Berichte“. Klicken Sie auf die Schaltfläche „Neu“, und wenn das Dialogfeld „Neuer Bericht“ angezeigt wird, wählen Sie den „Berichtsassistent“ und „qryFieldInfor“ aus dem Kombinationsfeld ganz unten im Dialog (siehe Abbildung 5.15).

Abbildung 5.15: Dieser Dialog wird verwendet, um den Report Wizard für unsere Abfrage zu starten.

Klicken Sie auf „OK“, um den Berichtsassistent für unsere Abfrage zu starten. Wählen Sie im ersten Bildschirm die Felder aus, die in unserem Bericht angezeigt werden sollen. In diesem Fall verwenden wir alle Felder in unserem Bericht. Um alle Felder auszuwählen, klicken Sie auf die Schaltfläche „>“. Klicken Sie auf „Weiter“, markieren Sie das Feld mit dem Namen FileHeading und klicken Sie auf die Schaltfläche „>“. Dieser Bildschirm, der in Abbildung 5.16 gezeigt wird, wird verwendet, um Gruppen für unseren Bericht zu definieren.

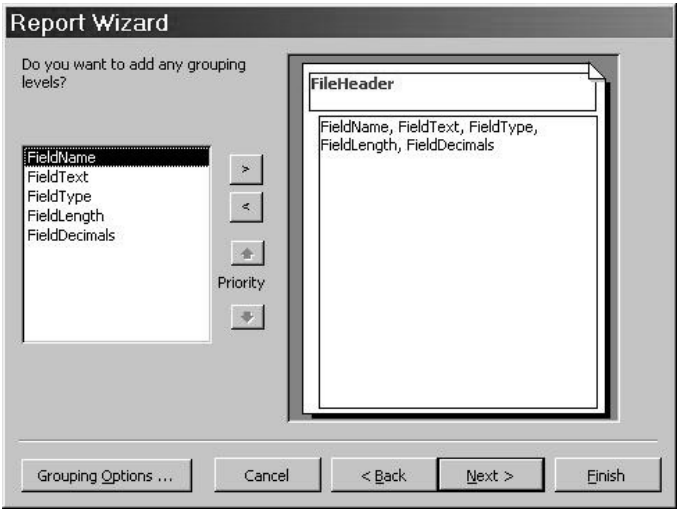


Abbildung 5.16: In diesem Dialog werden Berichtsgruppen definiert.

Wir erstellen das Datei-Heading-Feld für die Anzeige ganz oben in unserem Bericht. Dieses Feld ist eine Konkatenierung des Bibliotheknamens, Dateinamens und Dateitexts. Klicken Sie auf Weiter und verwenden Sie im Bildschirm für die Sortierung der Auswahl die erste Dropdown-Liste, um das Feld FieldName auszuwählen.

Der Berichts-Layout-Bildschirm in Abbildung 5.17 wird angezeigt.

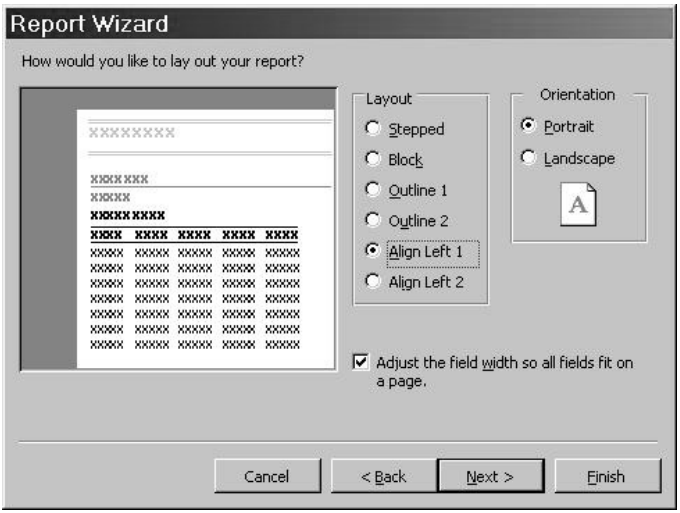


Abbildung 5.17: Der Berichts-Layout-Dialog ermöglicht es Ihnen, das Aussehen Ihres Berichts zu definieren.

Dieser Bildschirm ermöglicht es Ihnen, ein Layout aus verschiedenen Layouts zu wählen, um so das Aussehen Ihres Berichts festzulegen. Wählen Sie in unserem Beispiel Align Left 1 (links ausrichten). Klicken Sie noch einmal auf „Weiter“, um den letzten Dialog für den Assistent anzuzeigen. Ändern Sie den Berichtsnamen auf rptFieldInfo und aktivieren Sie die Optionsschaltfläche mit dem Namen Modify Report Design. Wenn der Report Designer, der in Abbildung 5.18 gezeigt wird, erscheint, klicken Sie auf die Seitenüberschrift rptFieldInfo und ändern den Text zu Feldinformationen. Ändern Sie ebenso die Überschrift für das Feld FileHeader auf Bibliothek/Datei. Wählen Sie im Menü „Datei speichern“, um diese Änderungen zu speichern.

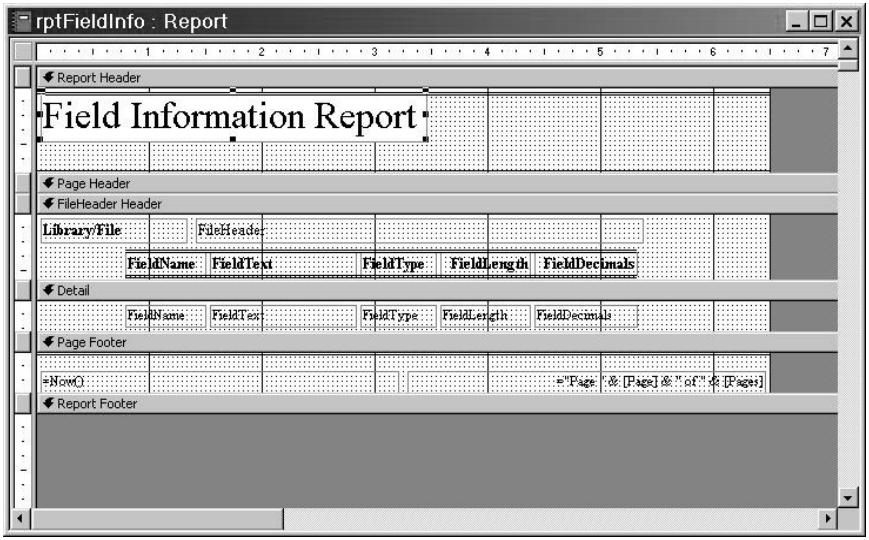


Abbildung5.18: Der Report-Designer wird verwendet, um Microsoft Access-Berichte zu erstellen und zu modifizieren.

Ob Sie es nun glauben oder nicht, unser Bericht ist fertig. Wir können die Anwendung testen, indem wir zum Datenbank-Fenster zurückkehren und auf das Register „Formulare“ klicken. Nun doppelklicken wir auf das Formular mit dem Namen frmReportPrompt. Wenn Sie noch nicht über eine aktive Client Access-Verbindung verfügen, werden Sie eventuell aufgefordert, Ihren Client Access-Benutzernamen und Ihr Passwort einzugeben. Anschließend wird der Bildschirm in Abbildung 5.19 angezeigt.

Um die Anwendung zu öffnen, wählen Sie einen Bibliotheksnamen aus dem Kombinationsfeld ganz oben im Fenster. Wenn die Dateiliste für diese Bibliothek angezeigt wird, doppelklicken Sie auf den gewünschten Dateinamen, um die Druckvorschau des Feld-

informationsberichts für diese Datei auszudrucken. Ein Beispiel für einen fertigen Bericht sehen Sie in Abbildung 5.20.

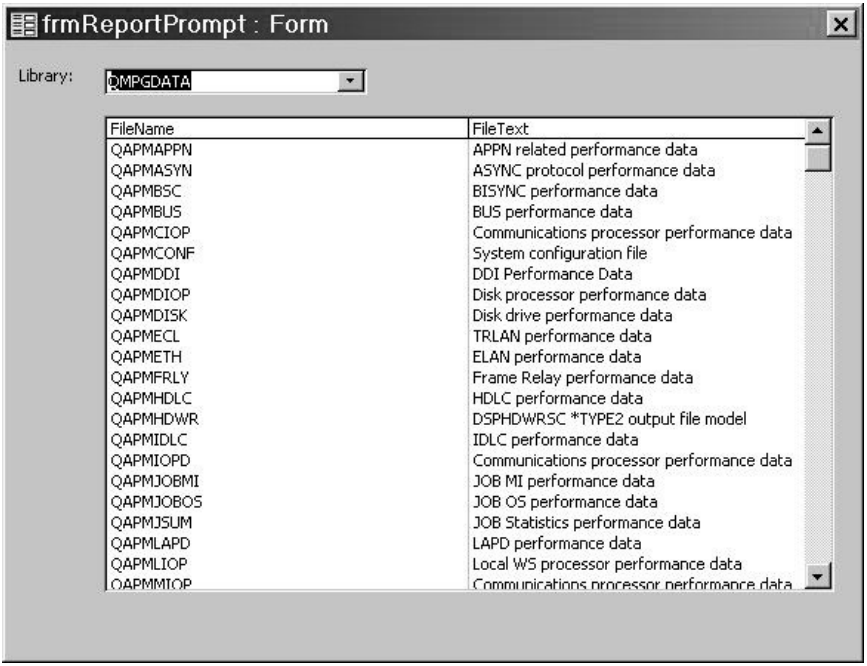


Abbildung 5.19: Die Bedienerführung für unsere Beispiel-ADO-Anwendung listet Dateien in einer angegebenen Bibliothek auf.

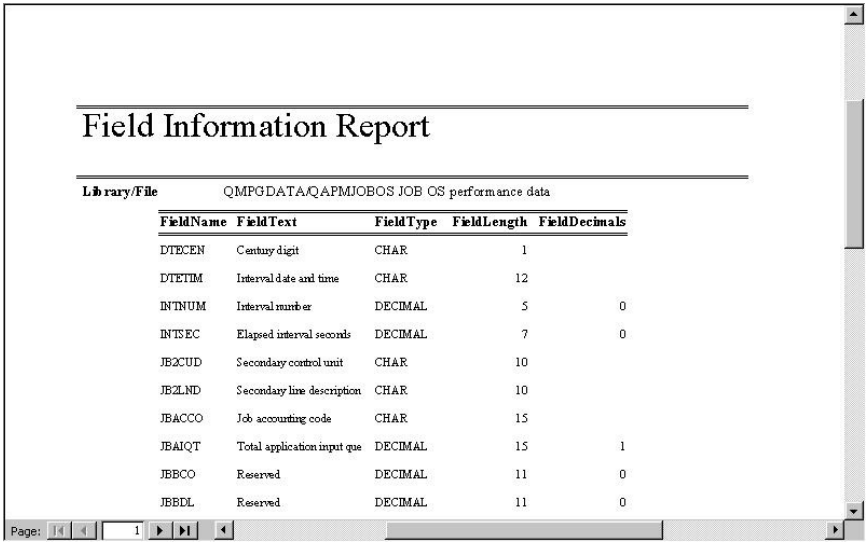


Abbildung 5.20: Der fertige Feldinformationsbericht in der Druckvorschau

Dieses Beispiel zeigt den grundlegenden Prozess, der verwendet wird, um einen Bericht zu erstellen, der es einem Benutzer ermöglicht, Parameter auf Grundlage von DB2/400-Daten anzugeben. Nun sehen wir uns an, wie ein Programm für die Aktualisierung von Daten mit Hilfe einer Visual-Basic-Anwendung erstellt wird.

Anwendung für die Dateiaktualisierung

Diese Anwendung basiert auf den gleichen Prinzipien wie unsere früheren Beispiele. Auch hier können wir SQL-Befehle für unsere DB2/400-Datenbank ausgeben. Das bedeutet, dass wir nicht nur eine SELECT-Anweisung für den Abruf von Daten verwenden, sondern wir können auch UPDATE eingeben, um Daten zu modifizieren, oder eine INSERT INTO Anweisung, um Daten hinzuzufügen.

In unserem Beispiel erstellen wir ein einfaches Programm, das eine Datei auf der 400 einliest und in diese Datei Daten schreibt. Hierfür erstellen Sie ein neues Standard-EXE-Visual-Basic-Projekt. In diesem Beispiel werden folgende vier Objekte verwendet:

1. Microsoft Flex Grid: Wird verwendet, um bereits eingegebene Datensätze anzuzeigen.
2. Text Box: Wird verwendet, um Daten für neue Datensätze einzugeben.
3. Label: Wird verwendet, um Überschriften zu erstellen.
4. Command Switch: Wird verwendet, um Datensatzaktualisierungen zu kontrollieren.

Abbildung 5.21 zeigt die Toolbox-Symbole, die für die einzelnen Objekte verwendet werden. Das Visual-Basic-Projekt ADOOrderEntry, das mit diesem Beispiel verknüpft ist, befindet sich auf der entsprechenden Webseite.

Sobald Sie das Projekt geladen haben, öffnen Sie das Formular frmOrderEntry, das in Abbildung 5.22 gezeigt wird. Diese An-

wendung aktualisiert Daten in DB2/400-Datenbankdateien. Sie enthält eine Auftragskopfdatei, die die Rechnungs- und Versandinformationen enthält, sowie eine Auftragsdetaildatei, die Auftragspositionen- und Auftragsmengeninformationen enthält.

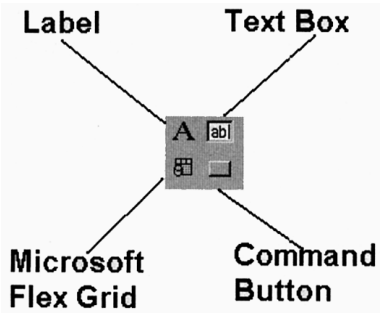


Abbildung 5.21:
Diese vier Objekte werden bei der Erstellung unserer Beispielanwendung verwendet.

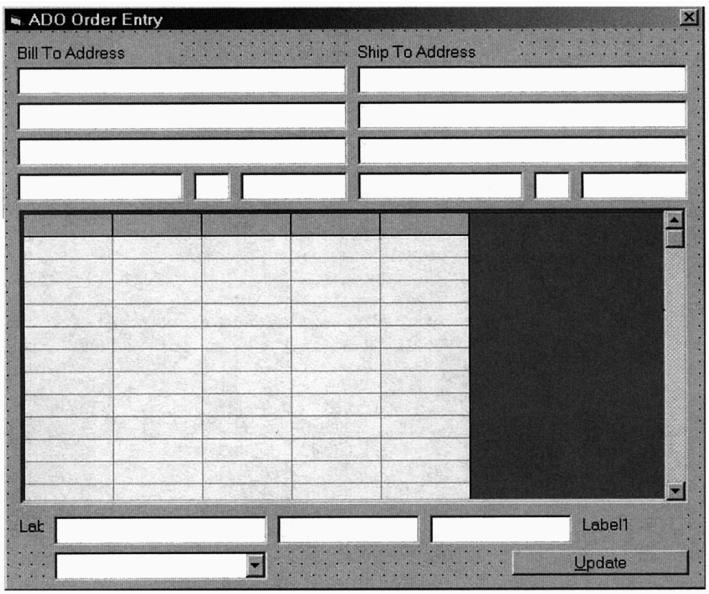


Abbildung 5.22: Das Hauptformular für das ADO-Programm für die Auftragserfassung

Nun zeigen wir das Form_Load-Event für unser Formular an, indem wir auf das Formular selbst doppelklicken. Abbildung 5.23 zeigt den Sourcecode für dieses Ereignis.

```
Private Sub Form_Load()
lblLineNo = " "
lblLineTotal = 0
fgdOrderLines.ColWidth(0) = lblLineNo.Width + 36
fgdOrderLines.ColWidth(1) = tbxLineDesc.Width + 36
fgdOrderLines.ColWidth(2) = tbxLineQty.Width + 36
fgdOrderLines.ColWidth(3) = tbxLinePrice.Width + 36
fgdOrderLines.ColWidth(4) = lblOrderTotal.Width + 36
fgdOrderLines.Rows = 2
fgdOrderLines.TextMatrix(0, 0) = "Ln"
fgdOrderLines.TextMatrix(0, 1) = «Description»
fgdOrderLines.TextMatrix(0, 2) = "Qty"
fgdOrderLines.TextMatrix(0, 3) = "Price"
fgdOrderLines.TextMatrix(0, 4) = "Total"

conn.ConnectionString = «Provider=IBMDA400; Data Source=90.0.0.3;»
conn.Open
On Error Resume Next
rsOrdHdr.Open "SELECT ORDNUM FROM OELIB.ORDHDR", conn
If conn.Errors.Count > 0 Then
   If conn.Errors(0).NativeError = 6036 Then
       cmd.ActiveConnection = conn
       cmd.CommandText = "CREATE COLLECTION OELIB"
       cmd.Execute
       cmd.CommandText = "CREATE TABLE OELIB.ORDHDR (ORDNUM NUMERIC " & _
       " (9, 0) NOT NULL WITH DEFAULT, BTNAME CHAR ( 30) NOT NULL " & _
       "WITH DEFAULT, BTADD1 CHAR ( 30) NOT NULL WITH DEFAULT, " & _
       "BTADD2 CHAR ( 30) NOT NULL WITH DEFAULT, BTCITY CHAR ( 20) " & _
       "NOT NULL WITH DEFAULT, BTSTTE CHAR (2 ) NOT NULL WITH DEFAULT " & _
       ", BTZIPC CHAR (10 ) NOT NULL WITH DEFAULT, STNAME CHAR ( 30) " & _
       "NOT NULL WITH DEFAULT, STADD1 CHAR ( 30) NOT NULL WITH DEFAULT" & _
       ", STADD2 CHAR ( 30) NOT NULL WITH DEFAULT, STCITY CHAR ( 20) " & _
       "NOT NULL WITH DEFAULT, STSTTE CHAR (2 ) NOT NULL WITH DEFAULT, " & _
       "STZIPC CHAR (10 ) NOT NULL WITH DEFAULT, ORDAMT NUMERIC (15 , 2) " & _
       "NOT NULL WITH DEFAULT) "
       cmd.Execute
```

Abbildung 5.23: Form_Load Event definiert unsere ADO-Verbindung und liest vorhandene Daten ein (Teil 1 von 2).

```
        cmd.CommandText = "CREATE TABLE OELIB.ORDDTL(ORDNUM NUMERIC " & _
        "(9 , 0) NOT NULL WITH DEFAULT, LINNUM NUMERIC (3 , 0) " & _
        "NOT NULL WITH DEFAULT, LINDES CHAR (35 ) NOT NULL WITH DEFAULT" & _
        ", LINQTY NUMERIC (9 , 0) NOT NULL WITH DEFAULT, LINPRC NUMERIC " & _
        "(13 , 4) NOT NULL WITH DEFAULT, LINTOT NUMERIC (14 , 2) " & _
        "NOT NULL WITH DEFAULT) "
        cmd.Execute

        rsOrdHdr.Open "SELECT ORDNUM FROM OELIB.ORDHDR", conn
    End If
End If
rsOrdHdr.MoveFirst
If rsOrdHdr.State 0 Then
Do Until rsOrdHdr.EOF

    cbxOrderNumber.AddItem rsOrdHdr.Fields("ORDNUM")

rsOrdHdr.MoveNext
Loop
End If
OrdNum = Val(cbxOrderNumber.List(cbxOrderNumber.ListCount - 1)) + 1
End Sub
```

Abbildung 5.23: Form_Load Event definiert unsere ADO-Verbindung und liest vorhandene Daten ein (Teil 2 von 2).

Dieses Event erledigt einige Arbeiten für Sie. Als Erstes passt es die Spaltenbreiten des Objekts Flex Grid an. Dann erstellt es eine ADO-Verbindung mit der DB2/400-Datenbank unter Verwendung des OLE DB-Providers von IBM. Anschließend versucht es, Auftragskopfdatensätze abzurufen, um das Kombinationsfeld cbxOrderNumber zu füllen. Wenn die Auftragskopfdatei nicht gefunden wird, werden ein ADODB-Befehlsobjekt und SQL CREATE-Anweisungen verwendet, um die Bibliothek, den Auftragskopf sowie Detaildateien zu erstellen, die von der Anwendung benötigt werden. In unserem Beispiel wurden Auftragspositionszeilen unter Verwendung der Textfelder ganz oben im Bildschirm eingegeben. Anschließend werden die fertiggestellten Zeilen in das Flex Grid eingefügt. Wenn Sie auf die Schalt-

fläche UPDATE klicken, werden die Auftragsinformationen aus dem Formular in die Dateien auf der 400 kopiert. Das cmdUpdate_click-Event in Abbildung 5.24 zeigt, wie dies geschieht. Dieses Event sendet auch mit QCMDEXC eine Nachricht in die QSYSOPR-Nachrichtenwarteschlange.

Dieser letzte Schritt illustriert, wie Sie ein Programm oder einen Befehl aufrufen können, indem Sie das ADO-Befehlsobjekt verwenden.

```
Private Sub cmdUpdate_Click()
Dim rs As New ADODB.Recordset, con2 As New ADODB.Connection

con2.ConnectionString = conn.ConnectionString
con2.Open
On Error Resume Next
 rsOrdHdr.MoveFirst
 rsOrdHdr.Find "ORDNUM=" & OrdNum, , adSearchForward

 If rsOrdHdr.EOF Then

cmd.ActiveConnection = con2
cmd.CommandText = "INSERT INTO OELIB.ORDHDR (BTNAME, BTADD1, BTADD2, BTCITY,
BTSTTE, " & _
                  "BTZIPC, STNAME, STADD1, STADD2, STCITY, STSTTE, STZIPC,
ORDAMT," & _
                  "ORDNUM) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
"
cbxOrderNumber.AddItem OrdNum
 Else
cmd.ActiveConnection = con2
cmd.CommandText = "UPDATE OELIB.ORDHDR SET BTNAME= ?, BTADD1= ?, BTADD2= ?,
BTCITY= ?, BTSTTE= ?, BTZIPC= ?, STNAME= ?, STADD1= ?, STADD2= ?, STCITY= ?,
STSTTE= ?, STZIPC= ?, ORDAMT= ? WHERE ORDNUM= ?"
End If
cmd.Parameters(0) = tbxBillName
cmd.Parameters(1) = tbxBillAdd1
```

Abbildung 5.24: Dieses Ereignis fügt Daten in OELIB auf der 400 hinzu und aktualisiert sie (Teil 1 von 3).

```
cmd.Parameters(2) = tbxBillAdd2
cmd.Parameters(3) = tbxBillCity
cmd.Parameters(4) = tbxBillState
cmd.Parameters(5) = tbxBillZip
cmd.Parameters(6) = tbxShipName
cmd.Parameters(7) = tbxShipAdd1
cmd.Parameters(8) = tbxShipAdd2
cmd.Parameters(9) = tbxShipCity
cmd.Parameters(10) = tbxShipState
cmd.Parameters(11) = tbxShipZip
cmd.Parameters(12) = lblTotal
cmd.Parameters(13) = OrdNum
Set rs = cmd.Execute
cmd.CommandText = ""
 If rsOrdHdr.EOF Then
 CmdStr = "'SNDMSG MSG(''Order # " & Str$(OrdNum) & " Added'')
TOUSR(QSYSOPR)'"
 CmdL = Format$(Len(CmdStr) - 4, «0000000000.00000»)
    cmd.CommandText = «QSYS.QCMDEXC(« & CmdStr & «,» & CmdL & «)»
    cmd.CommandType = adCmdStoredProc

    Set rs = cmd.Execute
    cmd.CommandText = ""
End If
cmd.CommandText = ""
If rsOrdDtl.State = 0 Then
rsOrdDtl.ActiveConnection = conn
rsOrdDtl.Open "SELECT * FROM OELIB.ORDDTL"
End If

For Z = 1 To fgdOrderLines.Rows - 1
 rsOrdDtl.Filter = "(ORDNUM=" & Str$(OrdNum) & ") AND (LINNUM = " & Str$(Z) &
")"
 cmd.CommandType = adCmdUnknown
  If rsOrdDtl.EOF Then
    cmd.CommandText = "INSERT INTO OELIB.ORDDTL  (LINDES, LINQTY, LINPRC,
LINTOT, ORDNUM, LINNUM) " & _
                      "VALUES(?, ?, ?, ?, ?, ?)"
    Else
    cmd.CommandText = "UPDATE OELIB.ORDDTL SET LINDES=?, LINQTY=?, LINPRC=?,
LINTOT=? WHERE ORDNUM=? AND LINNUM=? "
  End If
```

Abbildung 5.24: Dieses Ereignis fügt Daten in OELIB auf der 400 hinzu und aktualisiert sie (Teil 2 von 3).

```
If fgdOrderLines.TextMatrix(Z, 2)       "" Then
    cmd.Parameters(0) = fgdOrderLines.TextMatrix(Z, 1)
    cmd.Parameters(1) = fgdOrderLines.TextMatrix(Z, 2)
    cmd.Parameters(2) = fgdOrderLines.TextMatrix(Z, 3)
    cmd.Parameters(3) = fgdOrderLines.TextMatrix(Z, 4)
    cmd.Parameters(4) = OrdNum
    cmd.Parameters(5) = fgdOrderLines.TextMatrix(Z, 0)
Set rs = cmd.Execute
cmd.CommandText = ""

End If

Next
OrdNum = Val(cbxOrderNumber.List(cbxOrderNumber.ListCount - 1)) + 1
tbxBillName = ""
tbxBillAdd1 = ""
tbxBillAdd2 = ""
tbxBillCity = ""
tbxBillState = ""
tbxBillZip = ""
tbxShipName = ""
tbxShipAdd1 = ""
tbxShipAdd2 = ""
tbxShipCity = ""
tbxShipState = ""
tbxShipZip = ""
lblTotal = ""

For I = 1 To fgdOrderLines.Rows - 1
If fgdOrderLines.Rows > 1 Then
fgdOrderLines.RemoveItem 1
End If
Next

End Sub
```

Abbildung 5.24: Dieses Ereignis fügt Daten in OELIB auf der 400 hinzu und aktualisiert sie (Teil 3 von 3).

Wenn aus dem Kombinationsfeld ein Auftrag ausgewählt wird, kopiert das Ereignis cbxOrderNumber_Click in Abbildung 5.25 die Auftragsdaten aus der Datenbank zurück in das Formular.

```
Private Sub cbxOrderNumber_Click()

OrdNum = cbxOrderNumber
If rsOrdUpd.State 0 Then rsOrdUpd.Close

rsOrdUpd.Open "SELECT * FROM OELIB.ORDHDR AS A INNER JOIN " & _
                  " OELIB.ORDDTL AS B ON A.ORDNUM=B.ORDNUM " & _
                  " WHERE A.ORDNUM=" & cbxOrderNumber, conn


rsOrdUpd.MoveFirst
For I = 1 To fgdOrderLines.Rows - 1
If fgdOrderLines.Rows > 2 Then
fgdOrderLines.RemoveItem 1
End If
Next

tbxBillName = rsOrdUpd.Fields("BTNAME")
tbxBillAdd1 = rsOrdUpd.Fields("BTADD1")
tbxBillAdd2 = rsOrdUpd.Fields("BTADD2")
tbxBillCity = rsOrdUpd.Fields("BTCITY")
tbxBillState = rsOrdUpd.Fields("BTSTTE")
tbxBillZip = rsOrdUpd.Fields("BTZIPC")
tbxShipName = rsOrdUpd.Fields("STNAME")
tbxShipAdd1 = rsOrdUpd.Fields("STADD1")
tbxShipAdd2 = rsOrdUpd.Fields("STADD2")
tbxShipCity = rsOrdUpd.Fields("STCITY")
tbxShipState = rsOrdUpd.Fields("STSTTE")
tbxShipZip = rsOrdUpd.Fields("STZIPC")
```

Abbildung 5.25: Das cbxOrderNumber_Click-Ereignis wird verwendet, um DB2400-Daten in unserem Formular anzuzeigen (Teil 1 von 2).

```
Do Until rsOrdUpd.EOF
LineNo = rsOrdUpd.Fields("LINNUM")
If LineNo > 0 Then
    fgdOrderLines.AddItem " ", LineNo
    fgdOrderLines.TextMatrix(LineNo, 0) = LineNo
    fgdOrderLines.TextMatrix(LineNo, 1) = rsOrdUpd.Fields("LINDES")
    fgdOrderLines.TextMatrix(LineNo, 2) = rsOrdUpd.Fields("LINQTY")
    fgdOrderLines.TextMatrix(LineNo, 3) = rsOrdUpd.Fields("LINPRC")
    fgdOrderLines.TextMatrix(LineNo, 4) = Format$(rsOrdUpd.Fields("LINQTY")
* rsOrdUpd.Fields("LINPRC"), "  .00")
End If
rsOrdUpd.MoveNext
Loop

    lblOrderTotal = 0
For X = 1 To fgdOrderLines.Rows - 1
    lblOrderTotal = Val(lblOrderTotal) + Val(fgdOrderLines.TextMatrix(X, 4))
Next

End Sub
```

Abbildung 5.25: Das cbxOrderNumber_Click-Ereignis wird verwendet, um DB2400-Daten in unserem Formular anzuzeigen (Teil 2 von 2).

Drücken Sie die Taste F5, um die ADO-Anwendung für die Auftragseingabe zu kompilieren und auszuführen. Abbildung 5.26 zeigt die Formularansicht. Obwohl es sich hier um ein einfaches Beispiel handelt, haben Sie mit ihm alle Schritte kennen gelernt, die für die Erstellung einer Anwendung erforderlich sind, die aus Ihrer DB2/400-Datenbank Daten einliest und in diese Datenbank Daten schreibt.

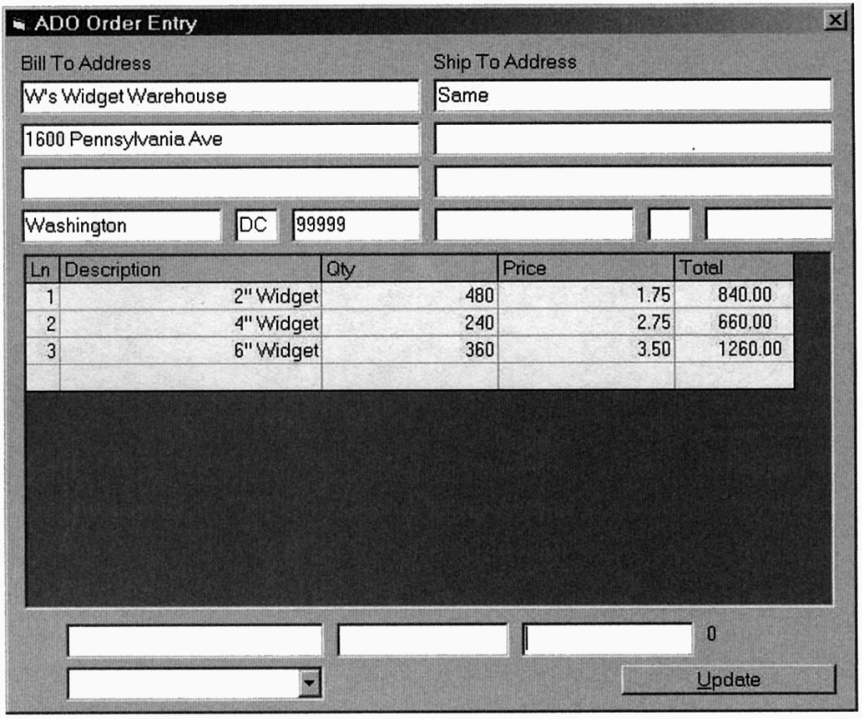


Abbildung 5.26: Das fertige ADO-Programm für die Auftragseingabe erstellt und aktualisiert DB2/400-Daten.

Wie Sie nun wissen, können Sie mit Hilfe von ActiveX-Datenobjekte und OLE DB problemlos PC-Anwendungen erstellen, die mit Ihrer DB2/400-Datenbank zusammenarbeiten. Zu den Vorteilen von ADO gehört es, dass Sie diese Techniken verwenden können, um auf andere Datenbanken – wie z.B. SQL Server oder Microsoft Access – zuzugreifen. Damit wird es ganz einfach, Daten von mehreren Datenbankplattformen in eine Anwendung zu integrieren.

6

FTP, Dateiübertragung und mehr

Sie wissen vermutlich bereits, wie ein File Transfer Protocol (FTP) Client für das Herunterladen von Dateien aus dem Internet verwendet wird. Was Sie aber wahrscheinlich nicht wissen, ist, dass Sie mit FTP noch viel mehr Aufgaben erledigen können, wenn Sie mit einem OS/400 FTP-Server verbunden sind. Die Bereitstellung von FTP durch OS/400 erlaubt es Ihnen nämlich, Programme aufzurufen oder Befehle auf Ihrer 400 von einer FTP Client-Sitzung aus auszuführen. Diese Fähigkeit kann in Situationen, bei denen ein Programm für die Ausführung erforderlich ist, bevor eine Datei übertragen wird oder nachdem eine Datei übertragen wurde, sehr nützlich sein. Wir befassen uns in diesem Kapitel mit der Frage, wie Sie diese Funktionalität optimieren können, um Ihnen dabei zu helfen, Ihre 400 mit PCs und LANs zu integrieren.

FTP-Grundlagen

Wie bei anderen FTP-Servern muss ein Benutzer bei der Anmeldung bei der OS/400-Implementierung von FTP einen Benutzernamen und ein Kennwort angeben. Jeder gültige OS/400-Anwender kann sich via FTP anmelden. Bevor Sie jedoch auf Ihre 400 via FTP zugreifen können, müssen Sie sicherstellen, dass der FTP-Server konfiguriert ist und ausgeführt wird. Verwenden Sie den Befehl Change FTP Attribute (CHGFTPA), um Ihre FTP-Serverkonfiguration anzuzeigen und – falls erforderlich – zu ändern. Abbildung 6.1 zeigt den Bildschirm, der angezeigt wird, wenn Sie CHGFTPA eingeben und F4 drücken, um die Bedienerführung aufzurufen.

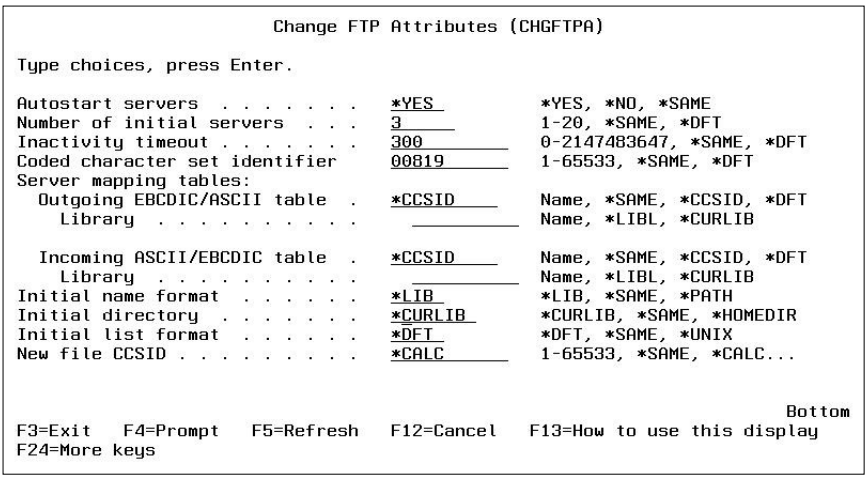


Abbildung 6.1: Der Befehl CHGFTPA wird verwendet, um den OS/400 FTP-Server zu konfigurieren.

Dieser Bildschirm wird verwendet, um den FTP-Server zu konfigurieren. Die erste Option definiert, ob der Server automatisch beim IPL ausgeführt wird. Der nächste Parameter definiert die Anzahl der Instanzen des FTP-Servers, die beim Start aktiviert werden. Der Parameter Inactivity Timer definiert die Zeit, die ein FTP-Client ohne Aktivität angemeldet bleiben kann, bevor der Server die Sitzung beendet. Coded Character Set Identifier legt fest, welche Codeseite von den einzelnen FTP-Clients verwendet werden soll. Die beiden nächsten Parameter bestimmen die ausgehenden und eintreffenden ASCII-zu-EBCIDIC-Übersetzungstabellen. Diese können für die Anpassung von Zeichenübersetzungen verwendet werden. Der Parameter Initial Name Format definiert das Datei-Namensformat, das verwendet wird, wenn ein Client sich zum ersten Mal anmeldet. Die Option *LIB sorgt dafür, dass Dateinamen im Format LIBRARY.FILE angezeigt werden. Das Format *PATH zeigt Dateien in Form ihres relativen Pfads. Der Parameter Initial Directory bestimmt, welche Bibliothek angezeigt wird, wenn sich ein Client anmeldet.

*CURLIB legt fest, dass die aktuelle Bibliothek für den Benutzer angezeigt wird. *HOMEDIR zeigt das Home-Verzeichnis des Benutzers an. Parameter Initial List Format definiert den Typ Dateiliste, der in einer Client-Sitzung angezeigt wird. *DFT verwendet das OS/400 Standard-Listing-Format. *UNIX zeigt die Dateiliste in einem ähnlichen Format wie im UNIX-Dateisystem an. Der Parameter New File CCSID schließlich definiert die Codeseite, die verwendet werden soll, wenn eine neue Datei von innerhalb einer FTP-Sitzung aus erstellt wird.

Wenn der FTP-Server nicht auf Autostart gesetzt wurde oder nicht ausgeführt wird, kann er mit Hilfe des Befehls Start TCP/IP Server (STRTCPSVR) gestartet werden:

```
STRTCPSVR SERVER(*FTP)
```

Nun können wir auf den OS/400-FTP-Server zugreifen. Als Erstes melden wir uns an, um sicherzustellen, dass keine Fehler aufgetreten sind. In Windows öffnen wir hierzu ein DOS-Fenster und geben FTP ein, gefolgt von der IP-Adresse Ihrer 400, und drücken die Eingabetaste. Sie werden dazu aufgefordert werden, einen Benutzernamen und ein Kennwort anzugeben. Sobald Sie einen gültigen Benutzernamen und Kennwort eingegeben haben, wird die FTP-Bedienerführung angezeigt, die Sie in Abbildung 6.2 sehen. Geben Sie DIR ein und drücken Sie die Eingabetaste, um eine Liste der Dateien im aktuellen Verzeichnis anzuzeigen:

```
C:\>ftp 198.162.0.1
Connected to 198.162.0.1
220-QTCP at S1234ABC.
220 Connection will close if idle more than 5 minutes.
User (90.0.0.3:(none)): QPGMR
331 Enter password.
Password:
230 QPGMR logged on.
ftp> dir
```

Abbildung 6.2: Ein Beispiel für die Anmeldung am FTP-Server von OS/400.

Der FTP-Befehl **remotehelp** wird verwendet, um eine Liste der Befehle anzuzeigen, die vom FTP-Server selbst unterstützt werden. Wenn Sie mit FTP nicht vertraut sind, finden Sie eine Liste der häufig eingesetzten FTP-Befehle und ihre Verwendung in Tabelle 6.1.

Tabelle 6.1: Einige häufig verwendeten FTP-Befehle	
Befehl(e)	**Beschreibung**
asc	Verwendung des ASCII-Übertragungsmodus.
bin	Verwendung des Binärübertragungsmodus.
cd	Änderung des aktuellen Verzeichnisses auf dem Server.
get, recv	Datei vom Server herunterladen.
lcd	Lokales Verzeichnis auf dem Client wechseln.
mkdir	Neues Verzeichnis auf Server erstellen.
put, send	Datei auf den Server hoch laden.
quote	Remote-Befehl an Server senden.

FTP-Scripting

Dateien können problemlos zur und von der 400 gesendet werden, indem die Befehle **recv** und **send** eingegeben werden. Vergessen Sie dabei nicht, dass die physischen Dateien, die Sie auf der 400 empfangen und senden, grundsätzlich Textdateien sind, wenn sie auf dem PC in Empfang genommen werden. Das bedeutet, dass alle Felder, die gepackte numerische Daten enthalten, nicht angezeigt werden können. Die Lösung für dieses Problem ist die Erstellung eines Programms für die Ausführung vor dem Download, das die Daten in eine Datei ohne gepackte Felder kopiert. Hier kommen Remote-Befehle ins Spiel. Verwenden Sie den FTP-Befehl **quote** FTP gemeinsam mit dem FTP-Server-Befehl RCMD, um ein Programm auf der 400 vom FTP-Programm aus auszuführen.

```
quote RCMD CALL PGM(XXX000CL)
```

In diesem Beispiel wird das Programm mit dem Namen XXX000CL über die FTP-Sitzung aufgerufen. Diese Funktion wird noch leistungsfähiger, wenn Sie eine Skript-FTP-Sitzung programmieren, die häufig verwendete Funktionen ausführt. Ein wichtiger Aspekt dabei ist die Tatsache, dass die Bibliotheksliste, die von der FTP-Sitzung verwendet wird, eventuell nicht den Anforderungen Ihres Programms entspricht. Das macht aber nichts, weil wir diese **quote** RCMD-Funktion verwenden können, um benötigte Bibliotheken zur Bibliotheksliste des aktuellen Jobs zu senden, bevor wir die Anwendung ausführen. Das in Abbildung 6.3 gezeigte Beispiel ergänzt die Bibliotheksliste des FTP-Jobs und ruft anschließend das benötigte Programm auf.

```
quote RCMD ADDLIBLE QFNTCPL
quote RCMD ADDLIBLE QSRV
quote RCMD ADDLIBLE MYLIB
quote RCMD CALL PGM(BLD001CL)
recv BLD001PF
quit
```

Abbildung 6.3: Eine Reihe von FTP-Befehlen, die für die Ausführung eines Programms vor dem Herunterladen einer Datei verwendet werden.

In diesem Beispiel wird der Befehl Add Library List Entry (ADDLIBLE) verwendet, um unsere Bibliotheksliste mit den Bibliotheken QFNTCPL, QSRV, und MYLIB zu ergänzen. Als Nächstes wird der Befehl CALL verwendet, um das CL-Programm BLD001CL auszuführen. Anschließend geben Sie den FTP-Befehl **recv** ein, um die Datei aus dem System herunterzuladen. Zuletzt beenden Sie die FTP-Sitzung mit dem Befehl **quit.** Sie können praktisch jeden OS/400-Befehl von innerhalb eines FTP-Skripts ausführen. Sie können ein Programm nicht nur einfach aufrufen, sondern Sie können auch z.B. Nachrichten an einen Benutzer senden oder eine Abfrage ausführen. Für die Zwecke der Fehlerbehebung und Fehlerbereinigung können Sie folgenden Befehl verwenden:

```
ftp> Quote RCMD DSPJOBLOG
```

Dieser Befehl sendet das Jobprotokoll für die FTP-Sitzung an den Standarddrucker. Durch Erstellung von *FTP-Scripting-Sitzungen* können Sie Funktionen für allgemeine Aufgaben erstellen. FTP-Skripte sind eigentlich Textdateien, die alle Tastenanschläge enthalten, die an den FTP-Server gesendet werden sollen. Dazu gehören der Benutzername und das Kennwort. Daher ist es normalerweise eine gute Idee,

ein Benutzerprofil zu erstellen, das eigens für FTP-Scripting vorgesehen ist. Damit erhält dieses Profil nur auf benötigte Objekte Zugriff. Wenn das Skript ausgeführt wird, wird die gesamte Ausgabe, die normalerweise in die FTP-Client-Anzeige gesendet wird, als QSYSPRT-Druckerdatei ausgegeben. Das ist auch ein großartiges Werkzeug für die Behebung von Problemen innerhalb eines FTP-Skripts.

Auf Client-Seite können Sie den Befehl „!“ verwenden, um eine DOS-Eingabeaufforderung anzuzeigen, oder auf diesen Befehl kann ein Programmname folgen, mit dem dieses Programm gestartet wird. Folgender Befehl startet z.B. den Windows Explorer und zeigt das Verzeichnis C:\Windows an:

```
ftp> ! EXPLORER C:\WINDOWS
```

Wenn ein DOS-Programm ausgeführt wird, wird die Verarbeitung des FTP-Skripts so lange angehalten, bis das DOS-Programm beendet wurde. Wenn ein Windows-Programm gestartet wird, setzt das FTP-Skript die Verarbeitung fort. Das ist ein wichtiger Punkt, den Sie nicht vergessen sollten, wenn die Fortsetzung der Verarbeitung des FTP-Skripts voraussetzt, dass eine gestartete Anwendung beendet wurde. Damit erhalten Sie die Möglichkeit, die gleiche Funktionalität auf Client-Seite auszuführen, die der Befehl RCMD auf Serverseite für Sie bereitstellt. Neben dem FTP-Befehl RCMD auf Serverseite werden noch mehrere weitere Befehle auf Serverseite vom OS/400 FTP-Server unterstützt. Eine teilweise Liste dieser Befehle sehen Sie in Tabelle 6.2. Dazu gehören Befehle für die Erstellung einer Bibliothek oder einer physischen Datei sowie Befehle, von denen Informationen über ein Objekt oder Objekte auf dem Server oder über den Server selbst bereitgestellt werden.

Tabelle 6.2: OS/400-Befehle FTP-Serverseite	
Befehl(e)	**Beschreibung**
ABOR	Beenden einer laufenden Datenübertragung.
ADDM FILE (Bibliothek/Datei) MBR(member)	Teildatei zu einer physischen Datei hinzufügen.
ADDV FILE (Bibliothek/Datei) MBR(member)	Teildatei mit variabler Länge physischer Datei hinzufügen.
APPE (Dateiname)	An Datei anhängen, falls vorhanden.
CDUP	Aktuelles Verzeichnis in Elternverzeichnis verschieben.
CRTL (Bibliothekname)	Bibliothek erstellen.
CRTP FILE (Bibliothek/Datei) RCDLEN(80) MAXMBRS(*NOMAX)	Physische Datei erstellen.
CRTS FILE (Bibliothek/Datei)	Physische Source-Datei erstellen .
CWD	Standardbibliothek erstellen.
DELE Bibliothek/Teildatei	Teildatei erstellen.
DLTF	Datei löschen.
DLTL	Bibliothek löschen.
LIST (Bibliothek/Teildatei)	Informationen über ein bestimmtes Objekt oder Objekte in einer angegebenen Datei anzeigen.

Tabelle 6.2: OS/400-Befehle FTP-Serverseite, Fortsetzung	
Befehl(e)	**Beschreibung**
MKD	Verzeichnis erstellen.
MODE (S=Stream, B=Block)	Modus für Datenübertragung angeben.
NOOP	Führt keine Funktion aus. Sendet nur als Antwort „OK“.
PWD	Gibt den Namen der aktuellen Bibliothek zurück.
QUIT	Beendet die FTP-Sitzung.
RCMD (OS400-Befehl)	Befehl auf Server ausführen.
REIN	Aktuelle Sitzung neu installieren.
RETR (Teildatei)	Daten aus angegebener Datei/Teildatei abrufen.
RMD	Verzeichnis entfernen.
RNFR von Name	Mit RNTO verwenden, um eine Datei umzubenennen. Gibt die umzubenennende Datei an.
RNTO zu Name	Mit RNFR verwenden, um eine Datei umzubenennen. Gibt den neuen Namen für die Datei an.
SITE (TRIM, LISTFMT, NULLSFLDS, CRTCCSID, NAMEFMT)	Mit angegebenen Parametern verwenden, um serverspezifische Optionen zu setzen.
STAT	Statusinformationen für die FTP-Sitzung.
STOR	Daten in Datei speichern und Inhalt ersetzen, wenn die Datei existiert.
STOU	Daten in Datei speichern und eindeutigen Namen verwenden, wenn die Datei existiert.
SYST	Namen des Remote-Betriebssystems zurückgeben.
TIME	Server-Timeout-Wert für diese FTP-Sitzung setzen.

Alle diese Befehle auf Serverseite können mit dem Befehl **quote** von einem FTP-Skript aus versendet werden. Sehen wir uns einmal den Prozess der Erstellung eines einfachen FTP-Skripts an. Starten Sie Windows Notepad, indem Sie im Startmenü auf „Ausführen“ klicken. Geben Sie nun NOTEPAD.EXE ein und drücken Sie die Eingabetaste. Wenn das Notepad-Fenster angezeigt wird, geben Sie den in Abbildung 6.4 gezeigten Code ein.

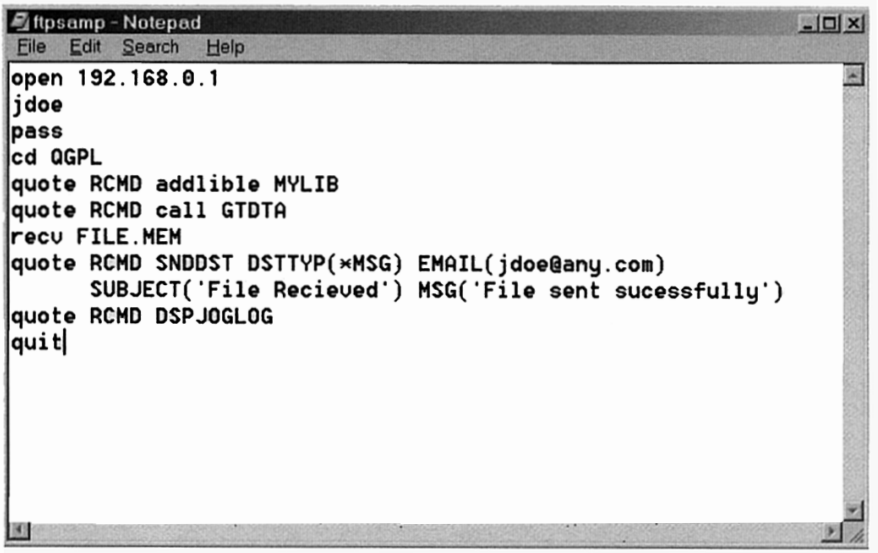


Abbildung 6..4: Dieses Beispiel verwendet den Subbefehl RCMD FTP für die Ausführung von OS/400-Befehlen.

In diesem Beispiel wird als Erstes die aktuelle Bibliothek in QGPL geändert, wobei der Befehl **cd** auf Client-Seite verwendet wird. Als Nächstes ergänzen wir die Bibliotheksliste für unsere FTP-Sitzung mit MYLIB. Anschließend rufen wir das Programm GTDTA auf und verwenden den Client-Befehl **recv**, um die ange-

gebene Datei herunterzuladen. Als Nächstes geben wir den OS/400-Befehl Send Distribution (SNDDST) ein, um eine e-Mail-Nachricht zu senden, die mitteilt, dass die Datei an einen angegebenen Benutzer versendet wurde. Als Letztes geben wir das Jobprotokoll aus, indem wir den OS/400-Befehl Display Job Log (DSPJOBLOG) eingeben, und beenden die FTP-Sitzung mit dem Client-Befehl **quit**. Speichern Sie diese Datei als FTPSAMP.SCP und beenden Sie Notepad. Nun rufen Sie ein DOS-Fenster auf und geben folgenden Befehl ein, um eine FTP-Sitzung mit Hilfe unserer Skriptdatei zu starten:

```
C:>FTP -s:ftpsamp.scp
```

Der Schalter „-s“ im FTP-Befehl wird verwendet, um anzugeben, dass der FTP-Client den Inhalt der angegebenen Datei senden soll, anstatt eine interaktive FTP-Bedienerführung anzuzeigen.

FTP-Scripting kann extrem nützlich sein, wenn Sie versuchen, eine PC-basierte Datenbank – wie z.B. SQL Server oder Microsoft Access – mit der 400 zu integrieren. Das FTP-Skript führt Befehle auf der 400 aus, die von einem Programm auf dem Client initiiert werden. So können Sie mit einem PC-Programm Daten aus einer SQL-Server-Datenbank in eine ASCII-Datei exportieren. Das Programm kann dann ein FTP-Skript initiieren, um die Datei zur 400 zu senden und ein CL-Programm aufrufen, um die Daten aus der ASCII Flat File in eine physische DB2/400-Datei zu kopieren. Ein weiteres Beispiel ist eine PC-Anwendung, die ein FTP-Skript ausführt, von dem ein 400-Programm aufgerufen wird, um bestimmte Datendateien zu erstellen – z.B. für die Tageseinnahmen – und dann die Daten in eine ASCII-Datei herunterzuladen. Anschließend importiert die PC-Anwendung diese Daten in eine

Microsoft Access-Datenbank, die für Verkaufsberichte verwendet wird.

Diese Aufgaben hätten zwar eventuell auch mit ODBC oder OLE DB erledigt werden können, aber das ist nicht immer möglich. Wenn Sie außerdem an die Möglichkeit denken, OS/400-Befehle auszuführen, werden Sie erkennen, dass FTP-Skript-Ausführung auf einem PC-Client äußerst nützlich sein kann.

Die iSeries 400 als FTP-Client

Die iSeries bietet nicht nur einen robusten FTP-Server, sondern stellt Ihnen auch einen FTP-Client zur Verfügung, der verwendet werden kann, um ähnliche Funktionen mit der 400 anstatt mit einem PC auszuführen. Der FTP-Client auf der 400 wird gestartet, indem in einer Befehlszeile FTP oder STRTCPFTP eingegeben wird – gefolgt von der IB-Adresse des FTP-Servers in einfachen Anführungszeichen. Abbildung 6.5 zeigt die Anzeige des OS/400 FTP-Clients.

Sicherlich auffallen wird Ihnen, dass die Benutzernamen-Eingabeaufforderung des FTP-Servers, wenn kein Benutzernamen eingegeben wird, den aktuellen Benutzernamen der 400 sendet. Die meisten der Client-Befehle, die von einem PC-Client unterstützt werden, werden auch von der 400 unterstützt. Der Befehl **help** kann verwendet werden, um den kompletten Hilfetext für den FTP-Client anzuzeigen.

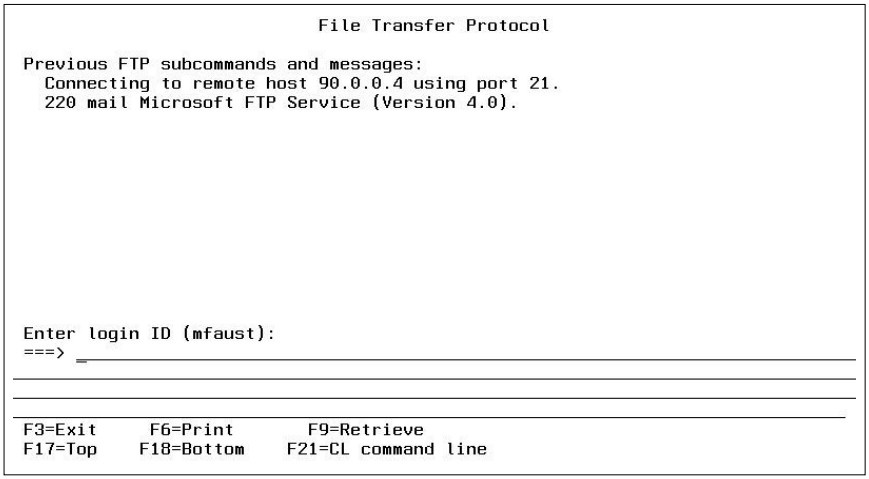


Abbildung 6.5: Die OS/400 FTP-Client Anmeldeanzeige

OS/400 FTP-Skripte werden entwickelt, indem zuerst eine Teildatei in einer physischen Sourcedatei erstellt wird, die alle Befehle enthält, die für die FTP-Sitzung erforderlich sind. Diesen Befehlen wird der Benutzername und das Passwort für die Anmeldung beim FTP-Server vorangestellt. Abbildung 6.6 zeigt ein FTP-Beispielskript, das sich am FTP-Server mit dem Benutzernamen ANONYMOUS anmeldet und dann das Passwort a@b.c. sendet (so wird die Anforderung der meisten FTP-Server erfüllt, gemäß der, dass an den FTP-Server eine e-Mail-Adresse als anonymes Passwort gesendet werden muss).

```
Anonymous
a@b.c
asc
send SALES.Jan
recv POS.txt POSDATA
quit
```

Abbildung 6.6: Ein Beispiel für das Aussehen eines OS/400 FTP-Skripts.

In der nächsten Zeile wird der FTP-Befehl **asc** eingegeben, um sicherzustellen, dass Übertragungen im ASCII-Modus erfolgen. In der nächsten Zeile wird der Befehl **send** verwendet, um die Datei mit dem Teildateinamen „Jan“ zum FTP-Server zu übertragen. Wie Sie sehen, wird auf die physische Datei im Format file.member verwiesen. In der nächsten Zeile verwenden wir den Befehl **recv,** um die Datei POS.txt in die physische Datei POSDATA herunterzuladen. Standardmäßig wird beim Herunterladen in eine vorhandene physische Datei die angegebene Teildatei oder die erste Teildatei, wenn keine angegeben wird, ersetzt. Der letzte Befehl im Skript lautet **quit**, der verwendet wird (wie

aus seiner Bezeichnung hervorgeht), um sich vom FTP-Server abzumelden.

Die Ausführung von FTP von einem Stapeljob aus erfolgt, indem Sie einen Befehl Override Database File (OVRDBF) für zwei Dateien, die vom Befehl **ftp** verwendet werden, ausführen. Die erste Datei mit der Bezeichnung INPUT speichert die Befehle für den Versand an den FTP-Server. Die zweite Datei erhält den Namen OUTPUT und wird nach Abschluss der FTP-Sitzung die Antworten enthalten, die vom FTP-Server zurück in die Sitzung gesendet wurden. Abbildung 6.7 zeigt ein Beispiel-CL-Programm, das verwendet wird, um das zuvor gezeigte FTP-Skript auszuführen.

```
 /*====================================================================*/
 /* Kompilierung:                                                      */
 /*                                                                    */
 /*             CRTCLPGM   PGM(XXX/FTP001CL) SRCFILE(XXX/QCLSRC)       */
 /*                                                                    */
 /*====================================================================*/
FTP001CL:       PGM

       OVRDBF        FILE(INPUT) TOFILE(MYLIB/FTPSCP) TOMBR(SENDDTA)
       OVRDBF        FILE(OUTPUT) TOFILE(MYLIB/FTPOUT) TOMBR(OUTPUT)

            FTP REMOTESYS(192.168.0.1)

            DLTOVR OVR(*ALL)

            ENDPGM
```

Abbildung 6.7: Dieses CL-Programm wird verwendet, um ein FTP-Skript von der 400 aus auszuführen.

In diesem Beispiel verwenden wir den CL-Befehl Override Database File (OVRDBF), um den Namen der Datei, der verwendet wird, um die Client-Befehle von unserem FTP-Skript aus zu

versenden, anzugeben. Der zweite OVRDBF-Befehl überschreibt mit der Ausgabedatei eine angegebene Datei. Diese Datei kann ein sehr wertvolles Debugging-Tool sein, wenn das CL-Programm, von dem der FTP-Befehl ausgeführt wird, die erwarteten Ergebnisse nicht erzielt. Denken Sie daran, dass Skripts wie diese von der Frage abhängig sind, ob die gesendeten Befehle erfolgreich ausgeführt werden. Genauer ausgedrückt, die Skripte ignorieren Befehle, die nicht erfolgreich ausgeführt werden.

Sie können nicht nur einfache Sende-/Empfangsskripte – wie beschrieben – erstellen, sondern auch komplexe FTP-Skripts, die viel mehr Aufgaben erledigen, als mit den PC-Clients möglich waren. Vorausgesetzt wird, dass der FTP-Server, mit dem Sie eine Verbindung aufnehmen, die entsprechende Funktion unterstützt – z.B. die Ausführung von Remote-Befehlen –, damit diese auf den Clients zur Verfügung stehen. Der FTP-Server von Microsoft z.B. unterstützt nicht die Ausführung von Remote-Befehlen. Sie erhalten diese Funktionalität jedoch, wenn Sie FTP-Server von Fremdanbietern einsetzen. Denken Sie auch daran, dass bei der Erstellung eines Skripts auf der 400, weil das Skript von einem CL-Programm ausgeführt wird, die gesamte Verarbeitung, die als Teil des Prozesses erfolgen muss, dem CL-Programm problemlos hinzugefügt werden kann. Damit können FTP-Skripte auf der 400 ähnliche Aufgaben übernehmen wie Skripte, die auf einem PC geschrieben wurden.

Fehlerbehandlung für FTP-Skripte

Fehlerbehandlung in einem traditionellem Sinn gibt es nicht innerhalb von FTP-Skripten, weil die Möglichkeit, bestimmte Zeilen eines FTP-Skripts abhängig von Bedingungen auszuführen, nicht gegeben ist. Sie können jedoch eine Anzahl von Techniken einsetzten, um Fehler zu behandeln, falls sie auftreten. Das wichtigste Werkzeug, das für die Fehlerbehandlung verwendet wird, ist die Datei OUTPUT, die erstellt wird, wenn das FTP-Skript ausgeführt wird. Der Nachteil dieser Methode ist, dass Sie Fehler nicht sofort behandeln können. Sie müssen sich dann mit ihnen befassen, nachdem Sie die Datei eingelesen haben und nachdem Sie nach zurückgekehrten Fehlern abhängig vom Antwortcode gesucht haben.

Ein RPG-Beispiel hierfür sehen Sie in Abbildung 6.8 gezeigt. Das Beispielprogramm liest die einzelnen Datensätze in der Datei OUTPUT ein und verwendet die Funktion %SCAN, um den Fehlercode 550 zu finden. Dieser wird zurückgegeben, wenn eine Datei für Versand oder Empfang nicht existiert. Diese Anwendung druckt dann ein Fehlerprotokoll für den Job aus. Aber Ihre Anwendung behandelt Fehler eventuell nicht auf die gleiche Weise und führt möglicherweise ein Programm aus, mit dem Daten erstellt werden bzw. es sendet eine Nachricht in die QSYSOPR-Nachrichtenwarteschlange.

```
 *=========================================================================
 *  Kompilierung:
 *         CRTBNDRPG PGM(XXX/SLG001RG) SRCFILE(XXX/QRPGLESRC)
 *
 *=========================================================================
 * FTP-Sitzung Ausgabedatei
FOUTPUT    IF   E            DISK
FQPRINTS   O    F   80       PRINTER OFLIND(*INOF)
 * Ausgabedatei einlesen bis Dateiende
C                   Eval     *INOF=*ON
C                   DoU      %EOF(OUTPUT)
C                   Read     OUTPUT
C                   If       %EOF(OUTPUT)=*OFF
C                   If       %Subst(SRCDTA:1:4)='550 '
C                   If       *INOF=*On
C                   Except   Header
C                   EndIf
C                   Except   Error
C                   EndIf
C                   EndIf
C                   EndDo
C                   Eval     *INLR=*ON
C                   Return
OQPRINTS   E            Header        2  2
O                                        30 'FTP Sitzung Fehlerbericht'
O          E            Error         1
O                       SRCDTA             80
```

Abbildung 6.8: Das RPG-Programm SLG001RG durchsucht eine FTP OUTPUT-Datei nach einem bestimmten Fehler.

Wenn Sie eine leicht modifizierte Version unseres vorherigen Beispiels kodieren, wird dieses Programm vom CL-Programm aus aufgerufen, mit dem das FTP-Skript gestartet wird. Abbildung 6.9 zeigt ein Beispiel für das Aussehen dieses Programms. Da der Override für die Datei OUTPUT bereits an Ort und Stelle ist, müssen wir nur noch unser Berichtsprogramm aufrufen.

Vergessen wir dabei nicht, dass wir in diesem Beispiel die gesamte Fehlerbehandlung nach dem Eintritt dieses Ereignisses ausführen. Dies kann sich auf die Art und Weise auswirken, in der Sie eine Anwendung entwickeln, die FTP-Skripte verwendet. Von diesem Punkt abgesehen, handelt es sich um ein leistungsfähiges Tool. Wenn Sie FTP-Sitzungen von einem iSeries-Stapeljob aus via Skript ausführen, können Sie Datenübertragungen ausführen, die von der iSeries aus gestartet werden – anstatt von einem PC aus.

```
 /*====================================================================*/
 /* Kompilierung:                                                      */
 /*                                                                    */
 /*        CRTCLPGM   PGM(XXX/FTP002CL) SRCFILE(XXX/QCLSRC)            */
 /*                                                                    */
 /*====================================================================*/
FTP002CL:  PGM

       OVRDBF    FILE(INPUT) TOFILE(MYLIB/FTPSCP) TOMBR(SENDDTA)
       OVRDBF    FILE(OUTPUT) TOFILE(MYLIB/FTPOUT) TOMBR(OUTPUT)

       FTP REMOTESYS(192.168.0.1)

       CALL SLG001RG

       DLTOVR OVR(*ALL)

       ENDPGM
```

Abbildung 6.9: Dieses CL-Programm führt ein FTP-Skript aus und druckt alle 550 empfangenen Fehler aus.

Die Entscheidung, ob Ihr FTP-Projekt auf der 400 oder auf einem PC ausgeführt werden soll, hängt weitgehend von den Anforderungen ab. Wenn z.B. Ihr FTP-Projekt es erfordert, dass Programme sowohl auf der 400 als auch auf dem PC ausgeführt werden, empfiehlt es sich eventuell, diese vom FTP-Client Ihres PCs aus auszuführen. Wenn der FTP-Job Teil eines vorhandenen Job-

Streams auf der 400 sein muss, macht es mehr Sinn, das FTP-Skript vom OS/400 FTP-Client aus zu starten. Gleichgültig jedoch, ob die Ausführung von einem PC-Client oder vom FTP-Client von OS/400 aus erfolgt: FTP kann viel mehr leisten, als nur Dateien übertragen.

Teil III
Cooles mit dem Internet

Im Laufe der letzten Jahre hat sich das Internet weiterentwickelt: vom Spielplatz für Technik-Freaks zu einem täglichen Arbeitsmittel für Unternehmen und Privatanwender. Heute ist das Internet ein wichtiger Bestandteil unseres Alltags – und die AS/400 unterstützt uns auch hier.

Die Möglichkeit, Daten Ihrer DB2/400-Datenbank zu hosten, kann viel Zeit sparen, weil Sie Ihre webbasierten Anwendungen schneller entwickeln können. Sie kennen sicherlich bereits den Websphere Web Server von IBM für die AS/400 und iSeries. Was Sie aber eventuell nicht wissen, ist, wie einfach Sie eine Webseite auf einem Microsoft NT Server erstellen können, auf dem ein Internet Information Server (IIS) ausgeführt wird, der Daten in eine DB2/400-Datenquelle einliest und Daten in diese Quelle schreibt. Sie werden erfahren, welche Schritte hierfür erforderlich sind und wie eindrucksvoll Webseiten erstellt werden können, die als Frontend für DB2/400 dienen. Sie wissen vielleicht auch nicht, dass OS/400 Dokumente an eine Internet-e-Mail-Adresse senden kann. Diese Funktion kann ein hervorragendes Tool für die Berichtverteilung sein. Wir beschäftigen uns in diesem Abschnitt mit der Frage, welche Konfiguration benötigt wird, um e-Mail-Nachrichten von der 400 zu senden. Wir werden eine Anwendung erstellen, mit der Sie Berichte von der 400 an eine Internet-e-Mail-Adresse als Textdatei, im Microsoft Word-kompatiblen RTF-Format oder sogar im HTML-Format senden können. Dieses Dienstprogramm ist von unschätzbarem Wert in Ihrem Software-Arsenal, und Sie werden sich sicherlich fragen, wie Sie bisher ohne ausgekommen sind.

Eine weitere aufkommende Internet-Technologie ist die Möglichkeit, sich an Ihrer AS/400 oder iSeries über das Internet anzumelden. So können Sie es z.B. entfernten Betriebsstätten oder mobi-

len Mitarbeitern erlauben, sich in Ihrem System anzumelden, ohne Verbindung mit Ihrem lokalen Netzwerk aufzunehmen. Wir werden uns einige Produkte ansehen, die Ihnen dies ermöglichen.

Zusätzlich werden wir weitere Methoden prüfen, um das Internet in Ihre 400 zu integrieren. Insgesamt erhalten Sie mit diesen Informationen alle Tools, die Sie benötigen, um die AS/400 und iSeries Internet-Konnektivität zu maximieren.

7

DB2/400-Webseiten auf IIS

Ein I.T.P.-Fachbuch

Obwohl die Entwicklung des robusten Webservers Websphere für AS/400 eine großartige Leistung von IBM war, ist das in vielen Fällen nicht die richtige Lösung. Ich möchte mich dabei klar ausdrücken: Es gibt keine bessere Plattform als die 400 für die Ausführung Ihrer geschäftskritischen Anwendungen. Aus genau diesem Grund aber empfiehlt es sich nicht, diese Plattform als Webserver zu verwenden. Eine hervorragende Alternative für diesen Anwendungstyp bietet ein Windows NT Server, auf dem Internet Information Server (IIS) ausgeführt wird. Wenn Sie IIS verwenden, können Sie Daten von Ihrer 400 für den Zugriff über das Internet bereitstellen, ohne Ihre 400 selbst ins Internet zu integrieren. Zusätzlich dazu vergrößern Sie nicht unnötig die Arbeitslast Ihrer 400. Stattdessen wird diese Arbeitslast auf kostengünstigere Windows NT-Servermaschinen verlagert. In diesem Kapitel wird untersucht, wie Webseiten auf IIS erstellt werden können, die in Ihrer DB2/400-Datenbank Daten erstellen und sogar updaten können.

DB2/400-Webberichte

Als Erstes erstellen wir eine Seite, die Informationen aus einer Datei auf der 400 in einer HyperText Markup Language- (HTML-) Tabelle anzeigt. Für unsere Beispiele erstellen wir Active Server Pages (ASP) mit VBScript. Diese Tools sind für die Erstellung von Seiten für den Datenzugriff hervorragend geeignet, weil sie so einfach zu handhaben sind. Die Erstellung von Webseiten, die auf DB2/400-Daten zugreifen können, könnte nicht einfacher sein. Der Dreh- und Angelpunkt bei Active-Server-Seiten ist die Tatsache, dass die Seite selbst auf dem Webserver ausgeführt wird und HTML erstellt, das an den Browser gesendet wird. Damit wird der Datenzugriffsabschnitt Ihres Programms völlig anonym, was bedeutet, dass der Endanwender nicht weiß, ob die Daten von einem SQL Server, DB2/400 oder einer anderen Datenbank eingelesen werden. Wir kennen bereits ADO und OLE DB für das Einlesen und Schreiben in die DB2/400-Datenbank von einem PC. Unsere Active Server Pages verwenden diese Technik mit Hilfe eines Visual Basic Scripts. Eine Active Server Page kann eine Kombination von HTML und anderen Skript-Sprachen – wie z.B. VBScript oder JavaScript – enthalten. VBScript-Zeilen können direkt in das HTML eingebettet werden, indem der Code innerhalb von <% und %> gesetzt wird.

Das Beispiel in Abbildung 7.1 zeigt eine Webseite, die VBScript enthält.

```
<HTML>
<HEAD>
<TITLE>Sample Page</TITLE>
</HEAD>

<BODY>
<% FOR X=1 to 20
    Response.Write(“Number=” & Str$(X))
   NEXT%>
</BODY>
</HTML>
```

Abbildung 7.1: VBScript kann in einer Active Server Page gemeinsam mit HTML verwendet werden.

In diesem Beispiel definieren wir den Seitentitel mit Hilfe von Standard-HTML-Tags. Als Nächstes verwenden wir eine VBScript FOR/NEXT-Schleife gemeinsam mit dem Befehl Response.Write, um den angegebenen Wert an einen Webbrowser zurückzugeben. Dieses einfache Beispiel – gemeinsam mit der OLE-DB – ist die Voraussetzung für die Erstellung einer Active Server Page, die auf die Daten auf der 400 zugreift.

Zuvor sehen wir uns die erforderliche Konfiguration an. Für die Erstellung von Active Server Pages für den DB2/400-Datenzugriff gelten zwei grundlegende Softwareanforderungen:

1. Microsoft Internet Information Server (IIS)

2. Client Access Express

Ein ODBC- oder OLE-DB-Treiber kann die zweite Softwarevoraussetzung ersetzen. Für unser Beispiel verwenden wir den OLE DB Provider von IBM (IBMDA400). Es empfiehlt sich auch, sicherzustellen, dass Sie die neueste Version der Data Access Component (MDAC) von Microsoft verwenden, die aus Systemkomponenten für ODBC, ADO, und OLE DB besteht. Prüfen Sie, ob Ihre Version aktuell ist, um Probleme im Vorfeld zu verhindern. Der Vorteil dabei ist, dass die OLE DB-Konnektivität auf dem Server erfolgt. D.h.: Es spielt keine Rolle, ob auf den Client-PCs die neueste Version von MDAC oder ein OLE DB-Treiber installiert ist.

In unserem ersten Beispiel erstellen wir eine Webversion des Feldinformationsberichts, der in Kapitel 5 mit Microsoft Access erstellt wurde. Für dieses Beispiel müssen zwei Active-Server-Seiten erstellt werden, wobei die erste Seite es dem Benutzer ermöglicht, die benötigte Bibliothek und die Datei auszuwählen. Die zweite Seite erlaubt es uns, die Felddaten anzuzeigen. Abbildung 7.2 enthält die Source für SAMPLE1.ASP.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Sample ASP Data Access Page 1</TITLE>
<H1>Datei-Feldinfo abrufen</H1>
</HEAD>
<BODY>
Wählen Sie einen Library-/Dateinamen aus dem Dropdown-Feld.</P>
<FORM ACTION="sample2.asp" METHOD="GET">
<SELECT NAME="libfile" SIZE="1" STYLE="WIDTH=360px">
```

Abbildung 7.2: SAMPLE1.ASPverwendet OLE DB für die Anzeige einer Liste von Bibliotheks- und Dateinamen für die Benutzerauswahl (Teil 1 von 2).

```
<%
   Set conn = Server.CreateObject("ADODB.Connection")
   on error resume next
    conn.open "Provider=IBMDA400;Data Source=S1234ABC;" & _
              "User ID=userid;Password=password;"
    sql = "SELECT TABLE_SCHEMA, TABLE_NAME FROM QSYS2.SYSTABLES" & _
    "ORDER BY TABLE_SCHEMA, TABLE_NAME"
    Set rs = Server.CreateObject("ADODB.Recordset")
    set rs.ActiveConnection=conn
    rs.cachesize = 32
    rs.Open sql
    for x=0 to conn.errors.count-1
    response.write conn.errors(x)
    next
    rs.MoveFirst
    if rs.EOF Then
    response.write "rs.eof"
    End If
    do until rs.eof
    response.write " " & rs.Fields("TABLE_SCHEMA") & _
    "." & rs.Fields("TABLE_NAME") & "</OPTION>"
    rs.MoveNext
    loop
    rs.close
    conn.close
%>

</SELECT>


<INPUT TYPE="submit" VALUE="Felder anzeigen">

</FORM>
</BODY>
</HTML>
```

Abbildung 7.2: SAMPLE1.ASPverwendet OLE DB für die Anzeige einer Liste von Bibliotheks- und Dateinamen für die Benutzerauswahl (Teil 2 von 2).

Der VBScript-Abschnitt auf dieser Seite erstellt zuerst ein OLE-DB-Verbindungsobjekt **conn**, das die Verbindung mit der DB2/400-Datenbank definiert. Denken Sie daran, dass – wie bei unseren früheren ADO-Beispielen sowie bei allen Beispielen in diesem Kapitel – die Datenquelle, die Benutzer-ID und die Passwortparameter mit Ihrem AS/400- oder iSeries-Systemnamen bzw. mit Ihrer IP-Adresse ersetzt werden müssen, gemeinsam mit einem gültigen Benutzernamen und einem Passwort für dieses System. Das ADO Recordset-Objekt wird anschließend verwendet, um den Bibliotheks- und die Dateinamen aus der Datei SYSTABLES in Bibliothek QSYS2 abzurufen. Diese Werte werden in ein Dropdown-Feld in einem HTML-Formular eingefügt, wobei der Befehl Response.Write verwendet wird, um die <OPTION> HTML-Tags auszugeben. Dieses Tag definiert individuelle Elemente in einem Listenfeld. Das Ergebnis ist ein Kombinationsfeld in einem HTML-Formular, das eine Liste aller Dateien und Bibliotheken im System im Format *library.file* enthält. Wenn ein Wert aus der Liste ausgewählt wird und der Benutzer auf die Schaltfläche „Felder anzeigen“ klickt, werden die Informationen an SAMPLE2.ASP übergeben, das die Feldinformationen anzeigt.

Abbildung 7.3 zeigt, wie SAMPLE1.ASP in Ihrem Browser aussehen sollte.

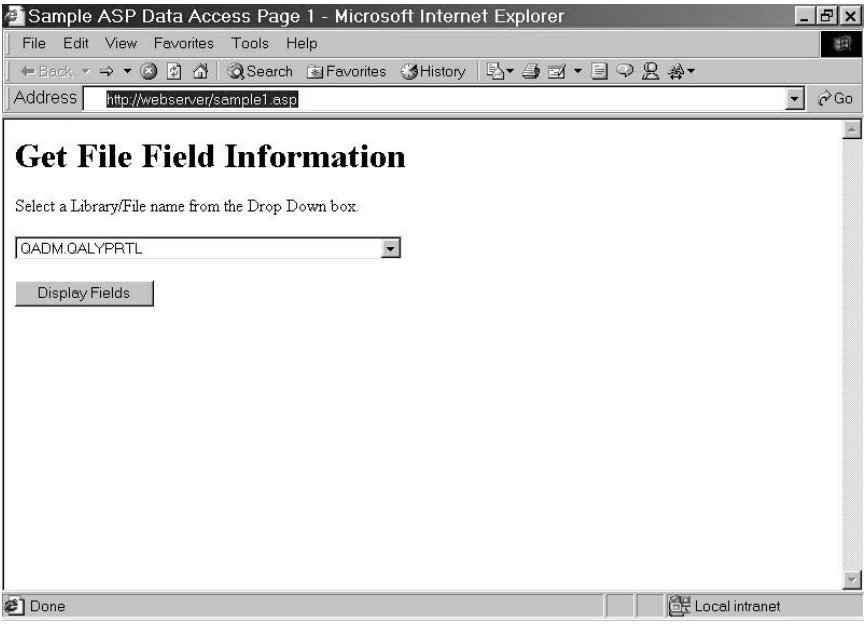


Abbildung 7.3: Diese Active Server Page verwendet OLE DB, um eine Liste der Bibliotheken und Dateien zu erstellen.

Wie Sie sehen, enthält das Kombinationsfeld eine Liste der Bibliotheks- und Dateinamen. Der ausgewählte Wert aus dieser Liste wird über die Variable LibFile in die Webseite eingegeben. Diese wird im Tag FORM definiert, wenn Sie auf die Schaltfläche Display Page klicken. Der Vorteil von Active Server Pages ist es, dass der VBScript-Code niemals zum Webbrowser gesendet wird. Nur HTML-Tags, die innerhalb des Scripts generiert werden – wie die Tags in unserem Beispiel für die Generierung der <OPTI-

ON> Tags –, werden im Browser angezeigt. Abbildung 7.4 zeigt ein Beispiel der HTML-Quelle, die zum Browser gesendet wird. In diesem Beispiel wurden einige der OPTION-Tags aus Platzgründen weggelassen.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Sample ASP Data Access Page 1</TITLE>
<H1>Get Dateifeldinformationen</H1>
</HEAD>
<BODY>
Select a Library/File name from the Drop Down box.</P>
<FORM ACTION="sample2.asp" METHOD="GET">
<SELECT NAME="libfile" SIZE="1" STYLE="WIDTH=360px">
… additional "<option>" tags omitted …
 QSVCDRCTR.QASDHOSTLF</OPTION>
 QSVCDRCTR.QASDMEDIA</OPTION>
 QSVCDRCTR.QASDPALEVT</OPTION>
 QSVCDRCTR.QASDPALPND</OPTION>
 QSVCDRCTR.QASDSACTY</OPTION>
 QSVCDRCTR.QASDSERIAL</OPTION>
 QSVCDRCTR.QASDTTDEV</OPTION>
 QSVCDRCTR.QASDTTPF</OPTION>
 QSVCDRCTR.QDDSSRC</OPTION>
 QSYS.QAANZLOG</OPTION>
 QSYS.QABNDBND</OPTION>
 QSYS.QABNDMBA</OPTION>
 QSYS.QABNDMCO</OPTION>
 QSYS.QABNDMEX</OPTION>
 QSYS.QABNDMIM</OPTION>
 QSYS.QABNDMPR</OPTION>
… additional "<option>" tags omitted …
</SELECT>


<INPUT TYPE="submit" VALUE="Felder anzeigen">

</FORM>
</BODY>
```

Abbildung 7.4: Dieser HTML-Code wird von unserer Active Server Page zum Browser gesendet.

Da nur HTML-Tags im Browser angezeigt werden, müssen Sie sich keine Sorgen darüber machen, dass jemand das Passwort zu sehen bekommt, das vom Skript verwendet wird, um auf die OLE DB-Datenquelle zuzugreifen. Der entsprechende Codeabschnitt wird auf dem Server ausgeführt und gelangt niemals zum Browser. Nun sehen wir uns den zweiten Bereich dieses Projekts an: die Seite, von der die Daten auf Feldebene angezeigt werden.

SAMPLE2.ASP – gezeigt in Abbildung 7.5 – enthält die Source für die Active Server Page, von der die Feldinformationsseite generiert werden wird. Diese Seite verwendet eine HTML-Tabelle für die Anzeige von Informationen zu jedem Feld in der ausgewählten Datei.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Sample ASP Data Access Page 2</TITLE>
<H1>Dateifeldinformationen</H1>
</HEAD>
<BODY>
<%
  Set conn = Server.CreateObject("ADODB.Connection")
  Lib=Left(Request.QueryString("LibFile"),InStr(1, _
           Request.QueryString("LibFile"),".")-1)
 File=Right(Request.QueryString("LibFile"), _
            Len(Request.QueryString("LibFile")) - _
            InStr(1,Request.QueryString("LibFile"),"."))

  on error resume next
   conn.open "Provider=IBMDA400;Data Source=S1234ABC;" & _
             "User ID=user;Password=password;"
   sql = "SELECT SYSTEM_COLUMN_NAME, COLUMN_TEXT, DATA_TYPE, LENGTH, " & _
```

Abbildung 7.5: SAMPLE2.ASP zeigt eine Liste von Feldern aus der ausgewählten Datei an (Teil 1 von 2).

```
          "NUMERIC_SCALE FROM QSYS2.SYSCOLUMNS " & _
          "WHERESYSTEM_TABLE_SCHEMA='"&Lib&"'ANDSYSTEM_TABLE_NAME='"&_
           File & "' ORDER BY SYSTEM_COLUMN_NAME"
   Set rs = Server.CreateObject("ADODB.Recordset")
   set rs.ActiveConnection=conn
   rs.cachesize = 32
   rs.Open sql
   If Conn.Errors.Count>0 Then
   for x=0 to conn.errors.count-1
   Response.write conn.errors(x)
   next
   Response.End
   End If
   Response.write "<table border='1' cellspacing='1' cellpadding='1'
width='80%'><thead><caption><FONT SIZE=-1 FACE='arial, helvetica'><b>Field
Information for File " & request.querystring("LibFile") & "</caption><tr>"
   For I=0 To rs.Fields.Count-1
   Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 FACE=" & _
                  "'arial, helvetica'><b><font color='#ffffff'>" & _
                  rs.Fields(I).Name & "</th>"
   Next
   Response.write "</tr>"
   Response.write "</THEAD>"
   Response.write "<TBODY>"

   rs.MoveFirst
   if rs.EOF Then
   Response.write "rs.eof"
   End If
   do until rs.eof
   Response.write "<tr>"
   For I=0 To rs.Fields.Count-1
   Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>" & _
                   rs.Fields(I) & "</td>"
   Next
   Response.write "</tr>"
   rs.MoveNext
   loop
   rs.close
   conn.close
%>

</BODY>
</HTML>
```

Abbildung 7.5: SAMPLE2.ASP zeigt eine Liste von Feldern aus der ausgewählten Datei an (Teil 2 von 2).

Die ausgewählte Bibliothek und Datei werden ganz oben auf unserer Seite angezeigt, wenn Sie den Befehl Response.Write gemeinsam mit dem Befehl Request.Querystring eingeben. Diese beiden Befehle werden in allen Beispielen in diesem Kapitel verwendet. Nehmen wir uns also eine Minute Zeit für sie. Das Objekt Response in VBScript wird verwendet, um die Ausgabe zum Client-Computer zu senden. Die Write-Methode dieses Objekts sendet die Ausgabe direkt in das aktuelle Browserfenster und macht es so möglich, den HTML-Code für Ihre Webseite von Ihrem VBScript aus dynamisch zu erstellen. Später werden wir auch die Redirect-Methode auf dieses Objekt anwenden, von dem die Steuerung von der aktuellen Seite zu einer neuen Seite verlagert wird. Das Objekt Request in VBScript ruft Informationen aus der aktuellen Sitzung ab. Die QueryString-Methode für dieses Objekt ruft Werte ab, die in diese Seite von einer vorherigen Seite innerhalb eines Abfragestrings gesendet wurde. Auf einer Webseite befindet sich der Abfragestring am Ende der URL mit vorangestelltem Fragezeichen (?). Diese Werte werden auch in die SQL-Anweisung SELECT übergeben. In allen Beispielen wird außerdem die Methode Form des Request-Objekts angewandt, die ebenfalls Werte aus dem aktuellen HTML-Formular abruft, jedoch ohne diese via QueryString zu übergeben. Das kann nützlich sein, wenn Sie Informationen übergeben möchten, die der Benutzer nicht sehen soll. Diese Seite verwendet eine sehr ähnliche Technik für den Datenabruf wie in SAMPLE1.ASP – mit einer Ausnahme. Dieses Beispiel positioniert die abgerufenen Daten in einer HTML-Tabelle unter Verwendung der <TD> und </TD> Tags. Diese Tags definieren die Daten für die Anzeige innerhalb der HTML-Tabelle. Abbildung 7.6 zeigt die resultierende Ausgabe von SAMPLE2.ASP.

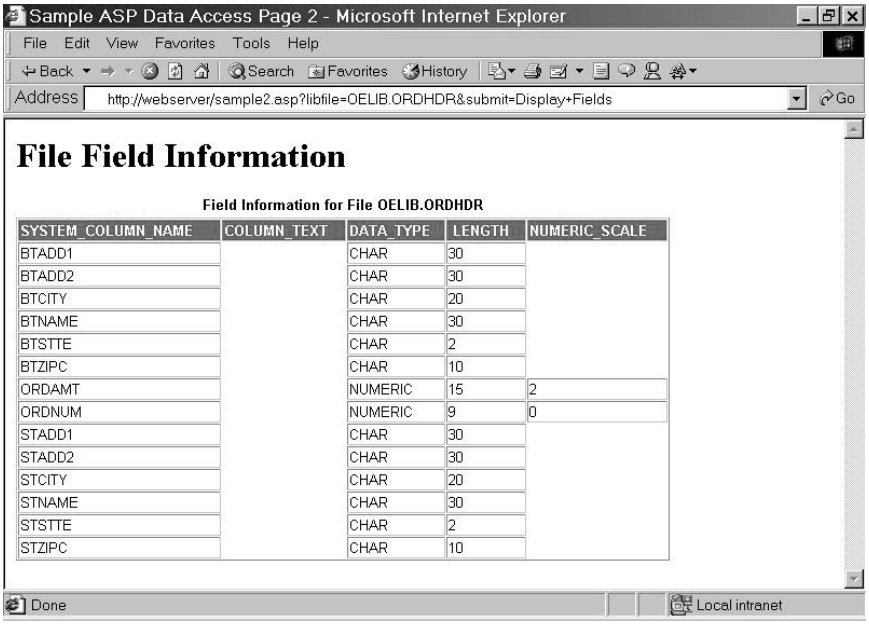


Abbildung 7.6: Die zweite Seite unserer Beispiel-ASP-Webseite zeigt Feldinformationen an.

Der große Vorteil dieser Technik ist, dass Sie sie wieder und wieder replizieren können. Sobald Sie einen Bericht mit einer Active Server Page erstellt haben, können Sie die SQL-Anweisung SELECT modifizieren, indem Sie Dateinamen, Feldauswahl oder WHERE-Klausel und die Heading-Informationen ändern, und in Kürze einen völlig neuen Bericht erstellen.

Als ob das noch nicht genug wäre, denken Sie daran, dass diese Seiten den Kunden oder Klienten über das World Wide Web zur Verfügung gestellt werden können. Sie können schnell und einfach eine Kundeninformations-Webseite erstellen, die Echtzeitdaten ent-

hält. Diese können auch als Alternative für die Erstellung von angepassten Berichten auf der 400 verwendet werden. Sie können attraktiv aussehende Berichte in einem Bruchteil der Zeit erstellen, die Sie für die Erstellung ähnlicher Berichte auf der 400 benötigen.

Aktualisierung von Daten auf einer Webseite

Nachdem wir eine Active Server Page erstellt haben, die Daten unserer DB2/400-Datenbank auf einer Webseite anzeigt, besteht der nächste Schritt darin, eine Active Server Page zu erstellen, die Daten in unserer Datenbank aktualisiert oder schreibt. Wie bei unserem letzten Beispiel weist der Prozess der Aktualisierung von Daten von einer Active Server Page große Ähnlichkeit mit dem Prozess auf, den wir verwenden, um dies von Access oder Visual Basic auszuführen, weil VBScripting die Grundlage unserer Active Server Page bildet. In unserem Beispiel erstellen wir eine neue Version der Visual Basic ADO-Anwendung für die Auftragserfassung, die in Kapitel 5 programmiert wurde. Diesmal verwenden wir die Active Server Page für die Anzeige und Aktualisierung unserer Daten. Abbildung 7.7 enthält die Source ORDERENT1.ASP, die die erste Seite unseres Beispiels ist und verwendet wird, um entweder einen Auftrag für die Aktualisierung auszuwählen oder um einen neuen Auftrag zu erstellen.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Sample ASP Order Entry Page 1</TITLE>
<H2>Select An Order Or Click "New Order"</H2>
</HEAD>
<BODY>
Order Number/Customer Name</P>
<FORM ACTION="orderent2.asp" METHOD="GET" id=form1 name=form1>
<SELECT NAME="ordno" SIZE="1" STYLE="WIDTH=360px">
```

Abbildung 7.7: Diese Seite ist der primäre Bildschirm für unsere webbasierte Anwendung für die Auftragserfassung (Teil 1 von 2).

```
<%
   Set conn = Server.CreateObject("ADODB.Connection")
   on error resume next
    conn.open "Provider=IBMDA400;Data Source=S1234ABC;" & _
              "User ID=user;Password=password;"
    sql = "SELECT ORDNUM, BTNAME FROM OELIB.ORDHDR ORDER BY ORDNUM"
    Set rs = Server.CreateObject("ADODB.Recordset")
    set rs.ActiveConnection=conn
    rs.cachesize = 32
    rs.Open sql
    for x=0 to conn.errors.count-1
    response.write conn.errors(x)
    next
    rs.MoveFirst
    do until rs.eof
    response.write " " & rs.Fields("ORDNUM") & " <tab> "& _
                   rs.Fields("BTNAME") & "</OPTION>"
    rs.MoveNext
    loop
    rs.close
    conn.close
%>

</SELECT>


<INPUT NAME="button" TYPE="submit" VALUE="Modify Order">
<INPUT NAME="button" TYPE="submit" VALUE="New Order">

</FORM>
</BODY>
</HTML>
```

Abbildung 7.7: Diese Seite ist der primäre Bildschirm für unsere webbasierte Anwendung für die Auftragserfassung (Teil 2 von 2).

Auf dieser Seite wird die gleiche Technik eingesetzt, die im vorherigen Beispiel verwendet wurde, um die Auftragsnummern in ein Kombinationsfeld zu laden. Die Seite liest die Auftragskopfdatei ein und erstellt ein <OPTION>-Tag für jede einzelne abge-

rufene Auftragsnummer. Abbildung 7.8 zeigt ein Beispiel für die resultierende Browseranzeige.

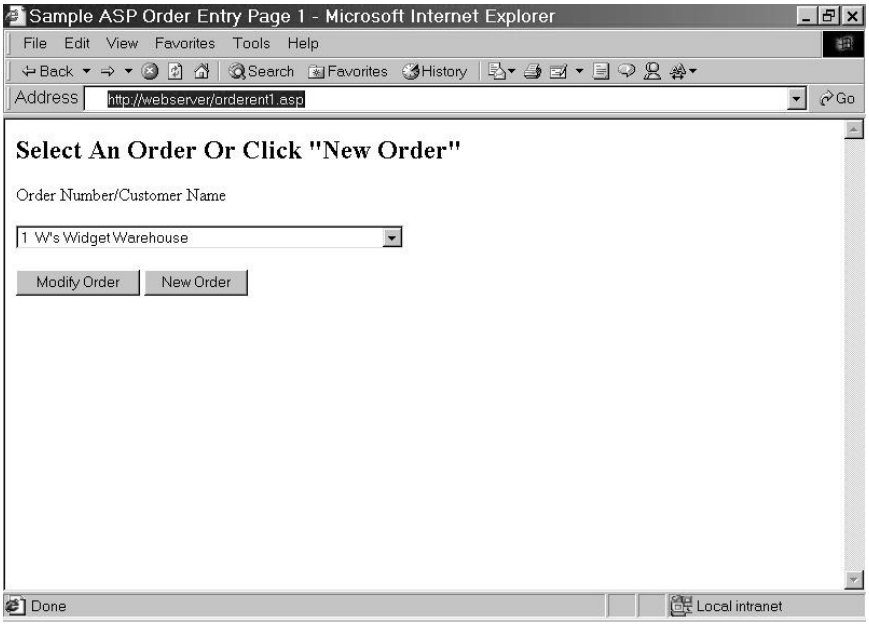


Abbildung 7.8: Dies ist die erste Seite unseres Programms für die Auftragserfassung.

Der Benutzer hat die Möglichkeit, einen vorhandenen Auftrag durch Auswahl eines Auftrags aus dem Kombinationsfeld zu aktualisieren und auf die Schaltfläche Modify Order zu klicken. Oder er erstellt einen neuen Auftrag, indem er einfach auf die Schaltfläche New Order klickt. Die Werte aus dem HTML-Formular werden an die zweite Seite unseres Beispiels übergeben, von der die Auftragsinformationen, die von der 400 abgerufen wurden, angezeigt werden. Er fügt die Daten in ein HTML-Formular ein,

um eine Modifizierung zuzulassen. Die Auftragsnummer wird über die Variable **ordno** übergeben, die Schaltfläche, auf die geklickt wird, wird mit Hilfe der Variable **button** identifiziert. Abbildung 7.9 zeigt die Source für ORDERENT2.ASP.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Sample ASP Order Entry Page 2</TITLE>
</HEAD>
<BODY>
<%
If Request.QueryString("button")="Modify Order" Then
    ordnum= Left(Request.QueryString("ordno"),   _
            InStr(1,Request.QueryString("ordno")," "))
    Response.Write "<H2><CENTER>Modify Order #:" & ordnum & "</H2>"

    Set conn = Server.CreateObject("ADODB.Connection")
    conn.open "Provider=IBMDA400;Data Source=S1234ABC;" & _
              "User ID =user;Password=password;"
    sql = "SELECT * FROM OELIB.ORDHDR AS H INNER JOIN OELIB.ORDDTL AS D " & _
          "ON H.ORDNUM = D.ORDNUM WHERE H.ORDNUM=" & _
          Ordnum & " ORDER BY H.ORDNUM, LINNUM"
    Set rs = Server.CreateObject("ADODB.Recordset")
    set rs.ActiveConnection=conn
    rs.cachesize = 96
    rs.Open sql,conn
    If conn.errors.count>0 Then
    for x=0 to conn.errors.count-1
        response.write conn.errors(x)
    Next    End If
    rs.MoveFirst
    If Not rs.EOF Then
    Response.write "<FORM ACTION='orderent3.asp' METHOD='Post'" & _
                   "id=form2 name=form2>"
    Response.write "<table border='0' cellspacing='1' cellpadding='1'" & _
                   "width='80%'>"
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>Bill To:</td>"
    Response.write "<td><input type=text name=btname size=30 maxlength=30 " & _
                  "value=" & Chr(34) & rs.Fields("BTNAME") &Chr(34) & "><br>"
```

Abbildung 7.9: Diese Seite aktualisiert Auftragsinformationen in Dateien auf der 400 (Teil 1 von 5).

```
    Response.write "<input type=text name=btadd1 size=30 maxlength=30 " & _
                   "value=" & Chr(34) & rs.Fields("BTADD1") & Chr(34) & "><br>"
    Response.write "<input type=text name=btadd2 size=30 maxlength=30 " & _
                   "value=" & Chr(34) & rs.Fields("BTADD2") & Chr(34) & "><br>"
    Response.write "<input type=text name=btcity size=20 maxlength=20 " & _
                   "Value=" & Chr(34) & rs.Fields("BTCITY") & Chr(34) & "> "
    Response.write "<input type=text name=btState size=2 maxlength=2 " & _
                   "Value=" & Chr(34) & rs.Fields("BTSTTE") & Chr(34) & ">"
    Response.write "<input type=text name=btzip size=10 maxlength=10 " & _
                   "Value=" & Chr(34) & rs.Fields("BTZIPC") & Chr(34) & _
                   "><br></td>"
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>Ship To:</td>"
    Response.write "<td><input type=text name=stname size=30 maxlength=30 " & _
                   "Value=" & Chr(34) & rs.Fields("STNAME") & Chr(34) & "><br>"
    Response.write "<input type=text name=stadd1 size=30 maxlength=30 " & _
                   "Value=" & Chr(34) & rs.Fields("STADD1") & Chr(34) & "><br>"
    Response.write "<input type=text name=stadd2 size=30 maxlength=30 " & _
                   "Value=" & Chr(34) & rs.Fields("STADD2") & Chr(34) & "><br>"
    Response.write "<input type=text name=stcity size=20 maxlength=20 " & _
                   "Value=" & Chr(34) & rs.Fields("STCITY") & Chr(34) & "> "
    Response.write "<input type=text name=stState size=2 maxlength=2 " & _
                   "Value=" & Chr(34) & rs.Fields("STSTTE") & Chr(34) & ">"
    Response.write "<input type=text name=stzip size=10 maxlength=10 " & _
                   "Value=" & Chr(34) & rs.Fields("STZIPC") & Chr(34) & _
                   "><br></td>"
    Response.write "<table border='1' cellspacing='1' cellpadding='1' " & _
                   "width='80%'><thead><caption><FONT SIZE=-1 FACE=" & _
                   "'arial, helvetica'><b> Order Line Details " & _
                   Request.QueryString("LibFile") & "</caption><tr>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 FACE=" & _
                   "'arial, helvetica'><b><font color='#ffffff'>Line #</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 FACE=" & _
             "'arial, helvetica'><b><font color='#ffffff'>Description</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
             "FACE='arial, helvetica'><b><font color='#ffffff'>Quantity</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
           "FACE='arial, helvetica'><b><font color='#ffffff'>Unit Price</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
           "FACE='arial, helvetica'><b><font color='#ffffff'>Line Total</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
            "FACE='arial, helvetica'><b><font color='#ffffff'>Delete</th>"
    Response.write "</tr>"
    Response.write "</THEAD>"
```

Abbildung 7.9: Diese Seite aktualisiert Auftragsinformationen in Dateien auf der 400 (Teil 2 von 5).

```
Response.write "<TBODY>"
Line = 0
Do Until rs.Eof
Line = Line +1
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>" & _
      rs.fields("LINNUM") &"<input type=hidden name=line" & trim(Line) & _
            " size=5 maxlength=5 value='" & rs.fields("LINNUM") & "'></td>"
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><input " & _
   "type=text name=item" & trim(Line) & " size=35 maxlength=35 value='" & _
    rs.Fields("LINDES") & "'></td>"
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><right>" & _
      "<input align=right type=text name=qty" & trim(Line) & _
      " size=15 maxlength=15 value='" & _
      FormatNumber(rs.Fields("LINQTY"),0) & "'></td>"
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><right>" & _
  "<input align=right type=text name=price" & trim(Line) & _
  " size=15 maxlength=15 value='" & FormatNumber(rs.Fields("LINPRC"),2) & _
  "'></td>"
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><right>" & _
                FormatNumber(Cdbl(rs.Fields("LINQTY")) * _
                Cdbl(rs.Fields("LINPRC")),2) & "</td>"
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><input " & _
               "type=checkbox name=del" & trim(Line) & "></td></tr>"
LinNum=Cint(rs.Fields("LINNUM"))+1
rs.MoveNext
Loop
rs.close
conn.close End If
Else
ordnum= CStr(Request.QueryString("NewOrd"))
Response.Write "<H2><CENTER>Create Order #:" & ordnum & "</H2>"

Response.write "<FORM ACTION='orderent3.asp' METHOD='Post' " & _
               "id=form2 name=form2>"
Response.write "<table border='0' cellspacing='1' cellpadding='1' " & _
               " width='80%'>"
Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>Bill To:</td>"
Response.write "<td><input type=text name=btname size=30 maxlength=30 " & _
               "value=''><br>"
Response.write "<input type=text name=btadd1 size=30 maxlength=30 " & _
               "value=''><br>"
Response.write "<input type=text name=btadd2 size=30 maxlength=30 " & _
               "value=''><br>"
```

Abbildung 7.9: Diese Seite aktualisiert Auftragsinformationen in Dateien auf der 400 (Teil 3 von 5).

```
    Response.write "<input type=text name=btcity size=20 maxlength=20 " & _
                   "Value=''> "
    Response.write "<input type=text name=btState size=2 maxlength=2 Value=''>"
    Response.write "<input type=text name=btzip size=10 maxlength=10 " & _
                   "Value=''><br></td>"
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>Ship To:</td>"
    Response.write "<td><input type=text name=stname size=30 maxlength=30 " & _
                   "Value=''><br>"
    Response.write "<input type=text name=stadd1 size=30 maxlength=30 " & _
                   "Value=''><br>"
    Response.write "<input type=text name=stadd2 size=30 maxlength=30 " & _
                   "Value=''><br>"
    Response.write "<input type=text name=stcity size=20 maxlength=20 " & _
                   "Value=''> "
Response.write "<input type=text name=stState size=2 maxlength=2 Value=''>"
    Response.write "<input type=text name=stzip size=10 maxlength=10 " & _
                   "Value=''><br></td>"
    Response.write "<table border='1' cellspacing='1' cellpadding='1' " & _
      "width='80%'><thead><caption><FONT SIZE=-1 FACE='arial, " & _
      "helvetica'><b> Order Line Details " & Request.QueryString("LibFile") & _
      "</caption><tr>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
             "FACE='arial, helvetica'><b><font color='#ffffff'>Line #</th>"
    Response.write “<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
           "FACE='arial, helvetica'><b><font color='#ffffff'>Description</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
            "FACE=’arial, helvetica'><b><font color='#ffffff'>Quantity</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
            "FACE=’arial, helvetica'><b><font color='#ffffff'>Unit Price</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
            "FACE=’arial, helvetica'><b><font color='#ffffff'>Line Total</th>"
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
            "FACE=’arial, helvetica'><b><font color='#ffffff'>Delete</th>"
    Response.write "</tr>"
    Response.write "</THEAD>"
    Response.write "<TBODY>"
    Line = 0
    LinNum=1

    End If
    Line = Line +1
```

Abbildung 7.9: Diese Seite aktualisiert Auftragsinformationen in Dateien auf der 400 (Teil 4 von 5).

```
    Response.write "
 </TR><td><FONT SIZE=-1 FACE='arial, helvetica'>" & _
          LinNum & "<input type=hidden name=line" & trim(Line) & " size=5 & _
          "maxlength=5 value='" & LinNum & "'></td>"
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><input " & _
        "type=text name=item" & LinNum & " size=35 maxlength=35 value=''></td>"
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><right>" & _
        "<input align=right type=text name=qty" & LinNum & " size=15 " & _
        "maxlength=15 value=''></td>"
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'><right>" & _
        "<input align=right type=text name=price" & LinNum & " size=15 " & _
        "maxlength=15 value=''></td>"
    Response.write "</TBODY></TABLE>"

    Response.write "<input TYPE='hidden' VALUE=" & ordnum &" name=ordno>"
    Response.write "
 <input TYPE='submit' VALUE='Update' id=submit1 " & _
                  "name=button1>"
    Response.write " <input TYPE='submit' VALUE='Cancel' name=button1></form>"

%>
</BODY>
</HTML>
```

Abbildung 7.9: Diese Seite aktualisiert Auftragsinformationen in Dateien auf der 400 (Teil 5 von 5).

Der VBScript-Abschnitt dieser Seite stellt als Erstes fest, auf welche Schaltfläche geklickt wurde, indem er den Befehl Retrieve.Querystring (bzw. Schaltfläche) abruft. Wenn die Schaltfläche New Order aktiviert wurde, ermittelt das Programm die Auftragsnummer, die für den neuen Auftrag verwendet werden soll, und zeigt einen leeren Bildschirm für die Auftragseingabe an. Wenn die Schaltfläche Modify Order aktiviert wurde, werden die Rechnungs- und Versandadressen gemeinsam mit allen Auftragspositionsdetails aus der DB2/400-Datenbank abgerufen. Diese Werte werden dann auf der Webseite angezeigt, um eine Modifizierung zu ermöglichen. Auf jeden Fall können Sie Positionen hinzufügen, in-

dem Sie Daten in die unterste Zeile des Detailpositionsabschnitts der Seite eingeben und auf die Schaltfläche Update klicken. Die Informationen, die Sie in diese Seite eingeben, werden in unserem Beispiel ORDERENT3.ASP an die nächste Seite übergeben. Diese Seite aktualisiert die Positionsdetails und die Auftragskopfinformationen mit den Änderungen, die auf der Seite eingegeben wurden. Über die Verknüpfung „Cancel“ – ganz unten im Bildschirm – gelangen Sie zurück in den ersten Bildschirm. Abbildung 7.10 zeigt die Ausgabe von ORDERENT2.ASP im Webbrowser.

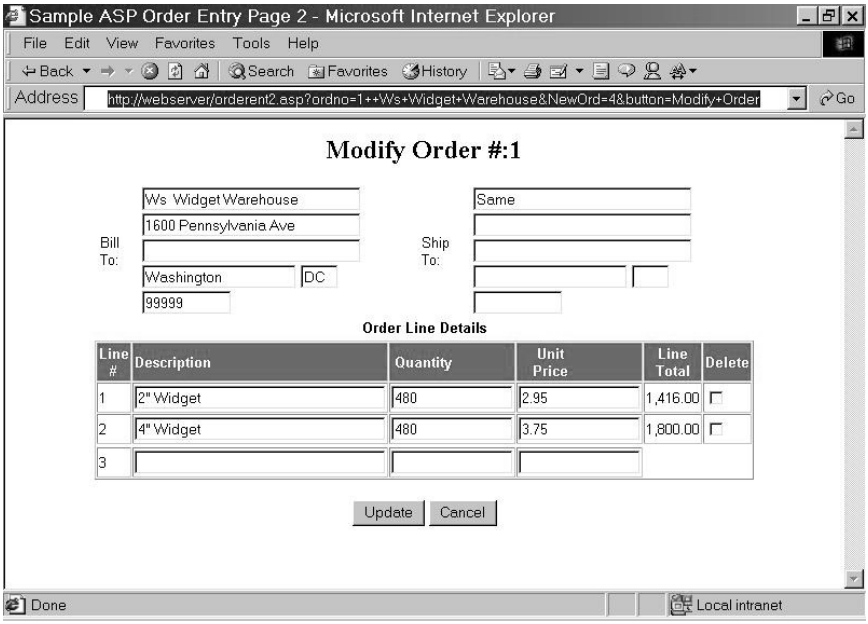


Abbildung 7.10: Eine webbasierte Anwendung für die Auftragserfassung.

Sicherlich auffallen wird Ihnen die Spalte ganz rechts in der Tabelle „Detailpositionen“, die mit Delete überschrieben ist und nur Kontrollkästchen enthält. Diese Spalte wird verwendet, um eine Position aus dem Auftrag zu löschen. Die Daten, die Sie in diese Seite eingeben, werden von der Active Server Page ORDERENT3.ASP aktualisiert. Diese Seite liest die Variablen aus dem HTML-Formular ORDERENT2.ASP und verwendet Werte, um Datensätze in den Dateien ORDHDR und ORDDTL in unserer DB2/400-Datenbank hinzuzufügen. Abbildung 7.11 enthält die Source für die ORDERENT3.ASP Active Server Page.

```
<%
 Line=1
 Total=0
 Do While Request.Form("line" & Trim(Line))""
 If IsNumeric(Request.Form("Qty" & Trim(Line))) Then
    Total=Cint(Total) + FormatNumber(Request.Form("Qty" & Trim(Line)) *
Request.Form("Price" & Trim(Line)),2)
 End If
    Line = Line + 1
 Loop
If Request.Form("button1")="Update" Then
 Set con2=Server.CreateObject("ADODB.Connection")

 con2.ConnectionString = "Provider=IBMDA400;Data Source=S1234ABC; " & _
                         "User ID=user;Password=password;"
 con2.Open
Set rs=Server.CreateObject("ADODB.Recordset")
 rs.open "SELECT * FROM OELIB.ORDHDR", con2
 rs.MoveFirst
 Ordno=Cint(request.form("ordno"))
 rs.Filter = "(ORDNUM=" & Cstr(request.form("ordno")) & ")"
 Set cmd=Server.CreateObject("ADODB.Command")

 If rs.EOF Then
 cmd.ActiveConnection = con2
 cmd.CommandText = "INSERT INTO OELIB.ORDHDR (BTNAME, BTADD1, BTADD2, " & _
                "BTCITY, BTSTTE, BTZIPC, STNAME, STADD1, STADD2, " & _
                "STCITY, STSTTE, STZIPC, ORDAMT, ORDNUM) " & _
                "VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
```

Abbildung 7.11: Diese Active Server Page aktualisiert unsere Datenbank (Teil 1 von 3).

```
 Else
 cmd.ActiveConnection = con2
 cmd.CommandText = "UPDATE OELIB.ORDHDR SET BTNAME= ?, BTADD1= ?, " & _
     "BTADD2= ?, BTCITY= ?, BTSTTE= ?, BTZIPC= ?, STNAME= ?, STADD1= ?, " & _

     "STADD2= ?, STCITY= ?, STSTTE= ?, STZIPC= ?, ORDAMT= ? WHERE ORDNUM= ?"
 End If
cmd.Parameters(0) = request.form("BtName")
cmd.Parameters(1) = request.form("BtAdd1")
cmd.Parameters(2) = request.form("BtAdd2")
cmd.Parameters(3) = request.form("BtCity")
cmd.Parameters(4) = request.form("BtState")
cmd.Parameters(5) = request.form("BtZip")
cmd.Parameters(6) = request.form("StName")
cmd.Parameters(7) = request.form("StAdd1")
cmd.Parameters(8) = request.form("StAdd2")
cmd.Parameters(9) = request.form("StCity")
cmd.Parameters(10) = request.form("StState")
cmd.Parameters(11) = request.form("StZip")
cmd.Parameters(12) = Total
cmd.Parameters(13) = request.form("ordno")
Set rs = cmd.Execute
cmd.CommandText = ""
Set rs1=Server.CreateObject("ADODB.Recordset")

rs.Open "SELECT * FROM OELIB.ORDDTL"
Line=1
Do While Request.Form("line" & Trim(Line))""

rs.Filter = "(ORDNUM=" & CStr(request.form("ordno")) & ") AND (LINNUM = " & _
            CInt(request.form("line" & trim(Line))) & ")"

 If rs.EOF Then
   cmd.CommandText = "INSERT INTO OELIB.ORDDTL   (LINDES, LINQTY, LINPRC, " & _

                     "LINTOT, ORDNUM, LINNUM) VALUES(?, ?, ?, ?, ?, ?)"
   Else

   If Request.Form("del" & Trim(Line))="on" Then
   cmd.CommandText = "DELETE FROM OELIB.ORDDTL WHERE ORDNUM=? AND LINNUM=? "
   Else
   cmd.CommandText = "UPDATE OELIB.ORDDTL SET LINDES=?, LINQTY=?, " & _
                     "LINPRC=?, LINTOT=? WHERE ORDNUM=? AND LINNUM=? "
   End If
End If
```

Abbildung 7.11: Diese Active Server Page aktualisiert unsere Datenbank (Teil 2 von 3).

```
If Request.Form("Qty" & Trim(Line)) "" Then
 If Request.Form("del" & Trim(Line))"on" Then
   cmd.Parameters(0) = Request.Form("Item" & Trim(Line))
   cmd.Parameters(1) = Request.Form("Qty" & Trim(Line))
   cmd.Parameters(2) = Request.Form("Price" & Trim(Line))
   cmd.Parameters(3) = FormatNumber(Request.Form("Qty" & _
         Trim(Line))*Request.Form("Price" & Trim(Line)),2)
   cmd.Parameters(4) = Request.Form("ordno")
   cmd.Parameters(5) = Request.Form("line" & trim(Line))
   Else
   cmd.Parameters(0) = Request.Form("ordno")
   cmd.Parameters(1) = Request.Form("line" & trim(Line))
 End if
   Set rs1 = cmd.Execute
   cmd.CommandText = ""

   End If
  Line=Line+1
  Loop

rs.close
con2.close
response.redirect "orderent2.asp?ordno=" & Request.Form("ordno") & "++" &
Request.Form("BtName") & "&button=Modify+Order"
Else
response.redirect "orderent1.asp"
End If
%>
```

Abbildung 7.11: Diese Active Server Page aktualisiert unsere Datenbank (Teil 3 von 3).

Diese Seite gibt keine Anzeige aus. Das VBScript auf dieser Seite erstellt als Erstes mit dem Befehl Server.CreateObjects unsere ADODB.Connection. Dieser Befehl wird auch verwendet, um das ADODB.Command-Objekt zu erstellen, mit dem SQL-Anwendungen ausgeführt werden, von denen Daten – einschließlich UPDATE und DELETE – modifiziert werden. Auffallen wird Ihnen an der ASP-Seite in Abbildung 7.11, dass diese nur VBScript-Code enthält. Die beiden anderen Seiten enthielten HTML und VBScript. Diese Seite

enthält kein HTML, weil sie nur für die Aktualisierung der Daten verwendet wird. Der Code prüft, ob der eingegebene Auftrag bereits existiert. Wenn nicht, wird ein SQL INSERT-Befehl als Befehlstext des Objekts cmd ADODB.command verwendet, um den neuen Datensatz hinzuzufügen. Wenn der Datensatz bereits existiert, wird der SQL-Befehl UPDATE verwendet, um Änderungen am Auftragskopf zu speichern. Diese Technik wird angewendet, um die Positionsdetails zu aktualisieren. Sobald die Dateiaktualisierungen fertiggestellt sind, wird die Steuerung an die Seite ORDERENT1.ASP oder ORDERENT2.ASP auf Grundlage der Schaltfläche übergeben, auf die der Benutzer auf der vorherigen Seite geklickt hat. Wenn das Löschkontrollkästchen Delete für eine Position aktiviert ist, wird eine DELETE-Anweisung für diese Position anstatt einer UPDATE-Anweisung ausgeführt. Der Befehl Response.Redirect sorgt dafür, dass die Steuerung entweder an die Webseite ORDERENT2.ASP oder an die Webseite ORDERENT1.ASP übergeben wird – basierend auf der Befehlsschaltfläche, auf die der Benutzer geklickt hat.

Diese drei Seiten sind eine einfache Anwendung für die Auftragserfassung. Was Ihnen jedoch eventuell auffallen wird, ist, dass ein Großteil des Codes, der hier verwendet wird, direkt von der Visual-Basic-Version dieser Anwendung übertragen wird, die in Kapitel 5 erstellt wurde. Das demonstriert die Portabilität von OLE DB für die Erstellung der Client-Server-Anwendung unter Verwendung der DB2/400-Datenbank. Großartig an der Active Server Pages ist, dass Sie nicht in eine Entwicklungsumgebung investieren müssen. Sie können einen einfachen Texteditor wie Windows Notepad verwenden, um die Source zu erstellen. Diese wird einfach auf Ihren IIS-Server kopiert, wenn er bereit für die Ausführung ist. Der Nachteil dieser Methode ist es, dass Ihnen keine Option für die Syntaxprüfung zur Verfügung steht, wenn

Sie eine Anwendung in Visual Basic entwickeln. Die Alternative besteht darin, den Microsoft Development Environment Editor einzusetzen, der eine optionale Komponente von Microsoft Office ist, oder den Microsoft Script Editor, den Sie von der Webseite von Microsoft herunterladen können. Beide Produkte bieten Ihnen die Funktionalität für die Syntaxprüfung von VBScript und Microsoft JavaScript. Microsoft Development Environment (MDE) ist dabei der etwas robustere Editor. Abbildung 7.12 zeigt die MDE-Anzeige.

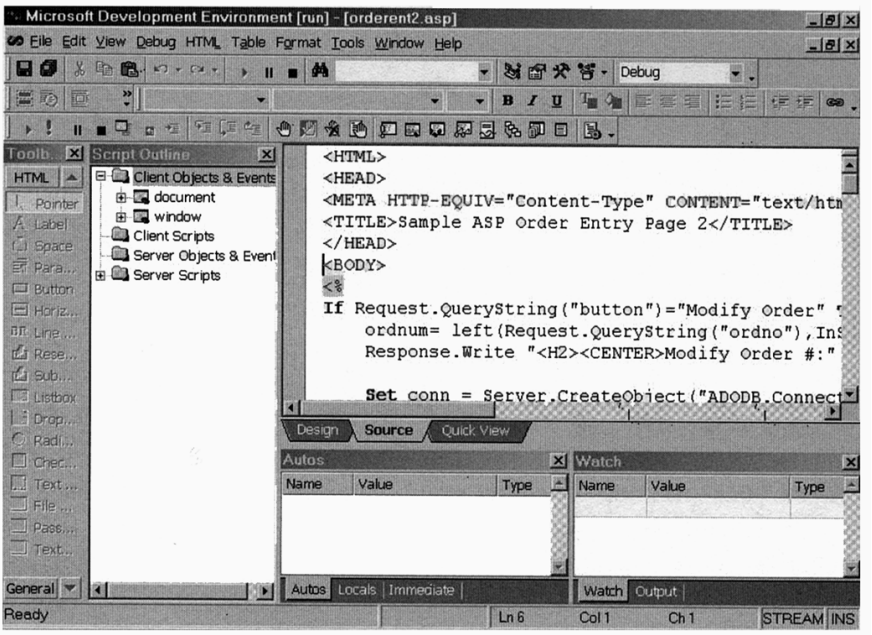


Abbildung 7.12: Der MDE-Editor kann benutzt werden, um ASP, HTML, Microsoft JavaScript und VBScipt zu bearbeiten.

Ein weiterer Nachteil bei der Entwicklung von Programmen mit Hilfe von Active Server Pages und VBScript ist das Debugging. Da der Code auf dem Server verwendet und nur HTML-Code zurück zum Client gesendet wird, kann das Debugging auf dem Client schwierig sein. Eine Technik besteht darin, Fehlerdaten zum Webbrowser-Client mit Hilfe des Befehls Response.Write zu senden. Diese Vorgehensweise empfiehlt sich, wenn Sie eine Vorstellung davon haben, an welcher Stelle der Fehler auftritt, jedoch den Grund dafür nicht kennen. Das ist beispielsweise der Fall, wenn in dem SQL-Code, den Sie für die Definition Ihres ADODB.Recordset-Objekts verwenden, ein Fehler auftritt und der Browser eine Meldung anzeigt, die Sie darüber informiert, dass der Server Fehler zurückgegeben hat. Diese Meldung ist jedoch nicht sehr aussagekräftig. Um mehr Informationen zum Problem zu erhalten, können Sie folgenden Code für den Befehl Recordset.Open eingeben:

```
.
.
On Error Resume Next

Rs.Open SQL,conn

If conn.errors.count>0 Then
   For X=0 to conn.errors.count-1
  Response.Write conn.errors(X)
   Next
End If
.
.
```

In diesem Beispiel wird als Erstes der Befehl On Error Resume Next verwendet, um die Weiterverarbeitung des Skripts zu ermöglichen, falls ein Fehler auftritt. Nach dem Befehl Rs.Open prüfen wir die Eigenschaft Count der Sammlung conn.errors, um festzu-

stellen, wie viele Fehler (falls überhaupt) vorhanden sind. Dieser Wert wird als der obere Bereich einer FOR/NEXT-Schleife verwendet, um die einzelnen existierenden Fehlermeldungen anzuzeigen. Wir wechseln von 0 zur Count-Eigenschaft -1, weil die Fehlersammlung für dieses Objekt ein nullbasiertes Array ist. Wenn Sie eine OLE DB-Verbindung mit einer DB2/400-Datenbank herstellen, enthält der dritte Fehler – conn.errors(2) – normalerweise eine Fehlernachricht, die auf die gleiche Weise formatiert ist wie die Anzeige der Nachricht im Jobprotokoll der 400. Mit dieser Technik werden alle diese Meldungen im Webbrowser angezeigt, womit das Debugging einfacher wird. Denken Sie dabei unbedingt daran, dass diese Meldungen, wenn der Befehl Rs.Open in einem Kombinationsfeld, Listenfeld oder in einer Tabellendefinition erscheint, eventuell nicht im Browser angezeigt werden. In diesem Fall müssen Sie den Code so ändern, dass die Meldung als Teil der Tabelle oder des Listenfelds angezeigt wird, indem Sie der Fehlernachricht das entsprechende HTML-Tag voranstellen. Oder Sie kommentieren den Bereich des Codes, von dem die Tabelle definiert wird, zeitweise aus. Diese Technik kann nicht nur für Debugging-Zwecke verwendet werden, sondern auch für die Fehlerbehandlung für Ihr fertiges Programm.

In unserem letzten Active-Server-Page-Beispiel erstellen wir eine leicht abgeänderte Version der Dateifeld-Informationsseite für die Erstellung eines einfachen Web-Abfragetools. Die erste Seite für dieses Beispiel – WEBQUERYMAIN.ASP – kann von der angegebenen Webseite heruntergeladen werden und ist nahezu identisch mit der Seite SAMPLE1.ASP, mit der wir zuvor gearbeitet haben. Diese Seite erstellt einfach ein Kombinationsfeld, von dem die Dateien von SYSTABLES im *Library-file*-Format aufgelistet werden. Die zweite Seite des Tools – WEBQUERYFIELDS.ASP

– zeigt die Feldliste in einer Tabelle gemeinsam mit Kombinationsfeldern an, die für die Auswahl der Felder für die Anzeige in unserer Webabfrage verwendet werden. Den Code für diese Seite sehen Sie in Abbildung 7.13.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Web Query Wizard</TITLE>
<CENTER><H1>File Field Selection</H1>
</HEAD>
<BODY>
<%
   Set conn = Server.CreateObject("ADODB.Connection")
   Lib=Left(Request.Form("LibFile"),InStr(1,Request.Form("LibFile"),".")-1)
   File=Right(Request.Form("LibFile"),Len(Request.Form("LibFile")) - _
         InStr(1,Request.Form("LibFile"),"."))

    conn.open "Provider=IBMDA400;Data Source=S1234ABC; " & _
              "User ID=user;Password=password;"
    sql = "SELECT SYSTEM_COLUMN_NAME, COLUMN_TEXT FROM QSYS2.SYSCOLUMNS " & _
          "WHERE SYSTEM_TABLE_SCHEMA='" & Lib & "' AND SYSTEM_TABLE_NAME='" & _
          File & "' ORDER BY SYSTEM_COLUMN_NAME"
    Set rs = Server.CreateObject("ADODB.Recordset")
    set rs.ActiveConnection=conn
    rs.cachesize = 32
    rs.Open sql
    If Conn.Errors.Count>0 Then
    for x=0 to conn.errors.count-1
    Response.write conn.errors(x)
    next
    Response.End
    End If
   Response.write "<FORM ACTION='Webquerysort.asp' METHOD='Post'  " & _
                  " id=form2 name=form2>"
   Response.write "<table border='1' cellspacing='1' cellpadding='1' " & _
                  "width='80%'>" & _
    "<thead><caption><FONT SIZE=-1 FACE='arial, helvetica'><b>Select " & _
                  "Fields from File " & request.Form("LibFile") & _
                  "</caption><tr>"
```

Abbildung 7.13: Diese Seite zeigt eine Liste von Feldern einer ausgewählten Datei für die Verwendung in einer Abfrage an (Teil 1 von 2).

```
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
                   " FACE='arial, helvetica'><b><font color='#ffffff'>" & _
                   "Select</th>"
    For I=0 To rs.Fields.Count-1
    Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
                   " FACE='arial, helvetica'>" & _
                   "<b><font color='#ffffff'>" & rs.Fields(I).Name & "</th>"
    Next
    Response.write "</tr>"
    Response.write "</THEAD>"
    Response.write "<TBODY>"

    rs.MoveFirst
    Line=0
    do until rs.eof
    Line=Line+1
    Response.write "<tr><td><FONT SIZE=-1 FACE='arial, helvetica'> " & _
                   "<input type=checkbox name=sel" & Trim(Line) & _
                   "><input type=hidden name=field" & Trim(Line) & _
                   " value='" & Trim(rs.Fields(0)) & "'></td>"
    For I=0 To rs.Fields.Count-1
    Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>" & _
                   rs.Fields(I) & "</td>"
    Next
    Response.write "</tr>"
    rs.MoveNext
    loop
    rs.close
    conn.close
 Response.Write "<INPUT TYPE='Hidden' size=63 name=LibFile Value = '" & _
             request.form("LibFile") & "'>

%>
</tbody></table>
WHERE:

<INPUT STYLE="WIDTH: 446px; HEIGHT: 146px" size=63 name=WHERECLAUSE>
<INPUT TYPE='submit' VALUE='Next' name=submit></form>
</BODY>
</HTML>
```

Abbildung 7.13: Diese Seite zeigt eine Liste von Feldern einer ausgewählten Datei für die Verwendung in einer Abfrage an (Teil 2 von 2).

Diese Seite verwendet eine Kombination von Techniken, die in den beiden letzten zwei Beispielen gezeigt werden. Die Felddaten werden auf die gleiche Weise wie in SAMPLE2.ASP abgerufen. Wir verwenden ein HTML-Formular, um Parameter zur nächsten Seite zu senden, einschließlich der Kontrollkästchen, die verwendet werden, um die Felder für die Anzeige in der fertigen Abfrage anzugeben. HTML-Formulare werden in ORDERENT2.ASP ebenfalls verwendet, wobei mit einem Textfeld als Teil des HTML-Formulars eine optionale WHERE-Klausel definiert wird (ohne das Wort WHERE), um die Datensätze auszuwählen, die in der fertigen Abfrage erscheinen sollen. Abbildung 7.14 zeigt die Webbrowser-Anzeige für diese Seite.

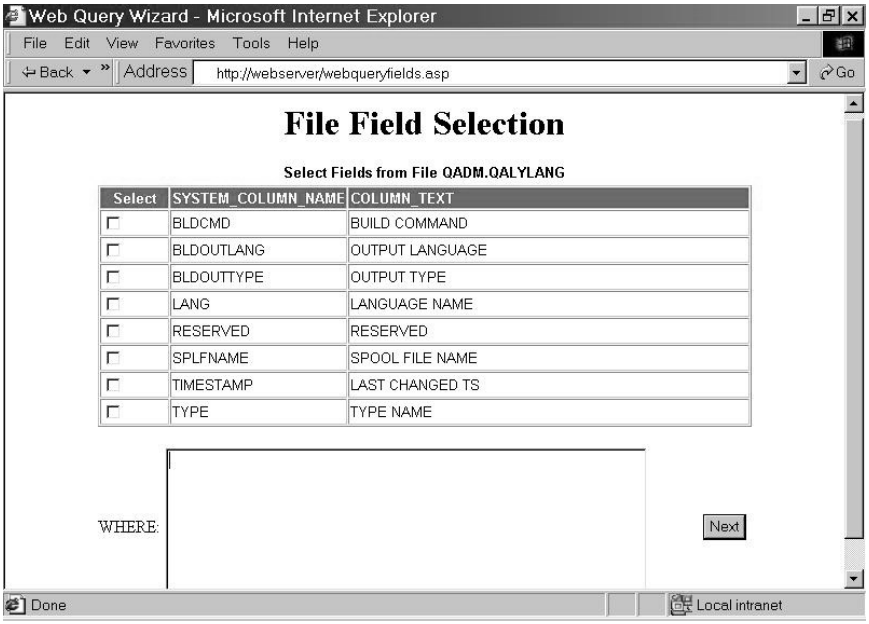


Abbildung 7.14: Diese Active Server Page wird verwendet, um Felder für unsere Abfrage auszuwählen.

Unsere nächste Seite – WEBQUERYSORT.ASP – zeigt eine Liste der Felder an, die im vorherigen Bildschirm ausgewählt wurden, um die Sortierreihenfolge mit Hilfe der Optionsschaltflächen innerhalb eines neuen HTML-Formulars zu definieren. Der Code für den Webbrowser-Client für diese Seite wird in Abbildung 7.15 gezeigt.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Web Query Wizard</TITLE>
<CENTER><H1>Sort Selection</H1>
</HEAD>
<BODY>
<%
   Line=1
   FldCnt=0
     Do While request.form("field" & Trim(Line))""
              If Request.Form("sel" & Trim(Line))="on" Then
                  FldCnt=FldCnt+1
              End If
     Line=Line + 1
     Loop
   Response.write "<FORM ACTION='Webquerysort.asp' METHOD='Post' " & _
                  "id=form2 name=form2>"
   Response.write "<table border='1' cellspacing='1' cellpadding='1' " & _
                  " width='80%'>" & _
                  "<thead><caption><FONT SIZE=-1 FACE='arial, " & _
                  "helvetica'><b>Define " & _
                  "Sort Order </caption><tr>"

   Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
                  "FACE='arial, helvetica'>" & _
                  "<b><font color='#ffffff'>Field</th>"

   For I=1 to FldCnt
   Response.write "<th BGCOLOR='#0066cc' WIDTH=20><FONT SIZE=-1 " & _
                  "FACE='arial, helvetica'>" & _
                  "<b><font color='#ffffff'>" & I & "</th>"
```

Abbildung 7.15: Diese Seite erstellt ein HTML-Formular für die Verwendung durch die Abfrage-Sortierdefinition (Teil 1 von 2).

```
    Next
    Response.write "</tr>"
    Response.write "</THEAD>"
    Response.write "<TBODY>"
    Line = 1
    Item = 0
      Do While request.form("field" & Trim(Line))""
              If Request.Form("sel" & Trim(Line))="on" Then
              Item=Item+1
              Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>" & _
       Request.form("field" & Trim(Line))
       Response.write "<INPUT TYPE=hidden NAME=Field" & Trim(Item) & _
       "Value='" & Request.form("field" & Trim(Line)) & "'></td>"
       For X=1 To FldCnt
               Response.write "<INPUT TYPE=radio NAME=sort" & trim(X) & _
               " Value='" & Item & "'"
       If X = Item Then
              Response.Write "CHECKED"
         End If
              Response.Write "></TD>"
       Next
              End If
       Response.Write "</TR>"
       Line=Line + 1
       Loop
       Response.Write "<INPUT TYPE='Hidden' size=63 name=LibFile Value = '" & _
               request.form("LibFile") & "'>
       Response.Write "<INPUT TYPE='Hidden' size=63 name=WHERECLAUSE Value =" & _
               Chr(34) & request.form("WHERECLAUSE") & Chr(34) & ">"
%>
</tbody></table>


<INPUT TYPE="submit" VALUE='Next' name=submit></form>


</BODY>
</HTML>
```

Abbildung 7.15: Diese Seite erstellt ein HTML-Formular für die Verwendung durch die Abfrage-Sortierdefinition (Teil 2 von 2).

Um ein HTML-Formular für die Definition der Sortierreihenfolge unserer Daten zu erstellen, entwickeln wir Optionsschaltflächen basierend auf der Anzahl der Felder für die Anzeige. Die Sortierreihenfolge wird basierend auf diesen Optionsschaltflächen bestimmt – ausgewählt von links nach rechts. Die Ausgabe sehen Sie in Abbildung 7.16.

Wir erstellen horizontale Gruppen von Optionsschaltflächen, die nur eine Sortierauswahl pro Feld zulassen. Wenn zwei Felder den gleichen Sortierwert aufweisen, erfolgt die Sortierung basierend auf dem Auftrag, in dem diese Felder in der Tabelle angezeigt werden. Zuletzt verwendet WEBQUERY.ASP die Informationen, die in den vorherigen Bildschirmen bereitgestellt wurden, um eine SQL-Anweisung zu generieren. Diese wird als Quelle für ein ADODB-Recordset verwendet, um Daten von der angegebenen Datenbankdatei abzurufen. Der Code für diese Seite wird in Abbildung 7.17 angezeigt.

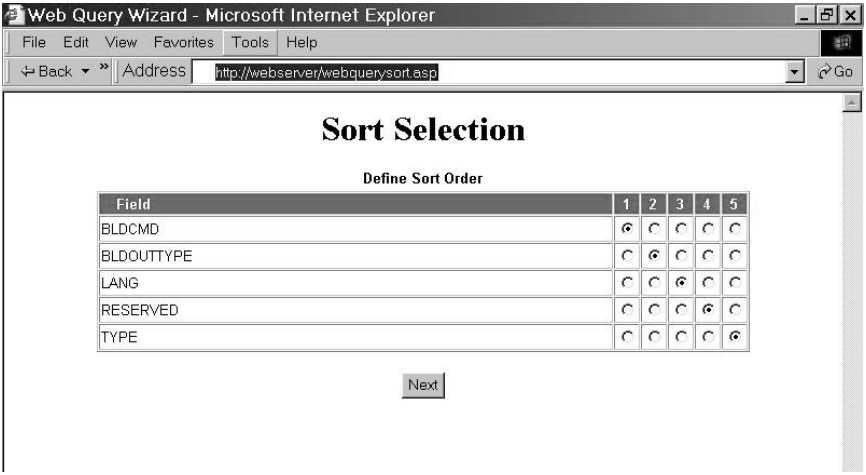


Abbildung 7.16: Dieser Bildschirm definiert die Sortierreihenfolge für die Daten in unserer Webabfrage.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=windows-1252">
<TITLE>Web Query Wizard</TITLE>
<CENTER><H1>Display Query</H1>
</HEAD>
<BODY>
<%
   Dim OrdBy()
   ReDim OrdBy(99)
   Set conn = Server.CreateObject("ADODB.Connection")
   Lib=Left(Request.Form("LibFile"),InStr(1,Request.Form("LibFile"),".")-1)
   File=Right(Request.Form("LibFile"),Len(Request.Form("LibFile")) - _
                InStr(1,Request.Form("LibFile"),"."))

'   on error resume next
    conn.open "Provider=IBMDA400;Data Source=S1234ABC; " & _
                "User ID=user;Password=password;"
    sql = "SELECT * FROM QSYS2.SYSCOLUMNS " & _
          "WHERE SYSTEM_TABLE_SCHEMA='" & Lib & "' AND SYSTEM_TABLE_NAME='" &

_
          File & "' ORDER BY SYSTEM_COLUMN_NAME"
    Set rs = Server.CreateObject("ADODB.Recordset")
    set rs.ActiveConnection=conn
    rs.cachesize = 32
    rs.Open sql
    If Conn.Errors.Count>0 Then
    For x=0 to conn.errors.count-1
    response.write conn.Errors(x) & "
"
    Next
    response.end
    End If
    Response.write "<table border='1' cellspacing='1' cellpadding='1' " & _
                "width='80%'><thead><caption><FONT SIZE=-1 FACE='arial, " & _

                "helvetica'><b>Query for File " & request.Form("LibFile") & _
                "</caption><tr>"
```

Abbildung 7.17: Dieser Code erstellt und gibt unsere komplette Webabfrage aus (Teil 1 von 3).

```
Line=1
Flds = " "

    Do While request.form("field" & Trim(Line))""
    If Flds" " Then Flds=Flds & ", "
    rs.Filter = "SYSTEM_COLUMN_NAME='" & Request.Form("field" & _
                Trim(Line)) & "'"
    rs.MoveFirst
    If Not rs.EOF Then
    If Len(rs.Fields("COLUMN_HEADING"))>0 Then
    Hdg=rs.Fields("COLUMN_HEADING")
Else
  Hdg=rs.Fields("SYSTEM_COLUMN_NAME")
End If
Response.write "<th BGCOLOR='#0066cc' WIDTH=60><FONT SIZE=-1 " & _
      "FACE='arial, helvetica'><b><font color='#ffffff'>" & Hdg & "</th>"
End If
Flds = Flds & Request.Form("field" & Trim(Line)) & " "
Ix=Request.Form("sort" & Trim(Line))
If Len(OrdBy(Ix))>0 Then
    OrdBy(Ix)=OrdBy(Ix) & ", "
End If
OrdBy(Ix)=OrdBy(Ix) & Request.Form("Field" & Trim(Line))
Line=Line+1
Loop
Srt = ""
For X=1 to Ix
If OrdBy(X)"" Then
If OrdrBy"" Then OrdrBy=OrdrBy & ", "
    OrdrBy=OrdrBy & OrdBy(X)
End If

Next
Sql = "SELECT " & Flds & " FROM " & Request.Form("LibFile")
If OrdrBy"" Then
      Sql = Sql & " ORDER BY " & OrdrBy
End If
Response.write "</tr>"
Response.write "</THEAD>"
Response.write "<TBODY>"
Set rs1 = Server.CreateObject("ADODB.Recordset")
```

Abbildung 7.17: Dieser Code erstellt und gibt unsere komplette Webabfrage aus (Teil 2 von 3).

```
    rs1.Open Sql,conn

    If Conn.Errors.Count>0 Then
    for x=0 to conn.errors.count-1
    Response.write conn.errors(x)
    next
    response.write Sql
    Response.End
    End If
    If rs1.EOF Then
              response.write "<td> No Records Returned </td></tr></Table>"
              response.end
      End If
    rs1.MoveFirst
    do until rs1.eof
    Response.write "<tr>"
    For I=0 To rs1.Fields.Count-1
        Response.write "<td><FONT SIZE=-1 FACE='arial, helvetica'>" & _
                    Trim(rs1.Fields(I).Value) & "</td>"
    Next
    Response.write "</tr>"
        rs1.MoveNext
    loop
    rs1.close
    conn.close

    Response.write "</TBODY></TABLE>"

%>

</BODY>
</HTML>
```

Abbildung 7.17: Dieser Code erstellt und gibt unsere komplette Webabfrage aus (Teil 3 von 3).

Auf dieser Seite werden zwei separate OLE DB-Recordsets verwendet, um zuerst die Spaltenüberschriften für die ausgewählten Felder in unserer Tabelle abzurufen und anschließend die tatsächlichen Daten. Die SQL-Anweisung für das zweite Recordset wird basierend auf den Informationen erstellt, die die letzten drei

Webseiten bereitstellen. Wenn ein Fehler auftritt, verwenden wir die bereits beschriebene Fehlerbehandlungstechnik für die Anzeige einer Nachricht im Browserfenster, von der das Problem beschrieben wird. Abbildung 7.18 zeigt ein Beispiel für diese Ausgabe.

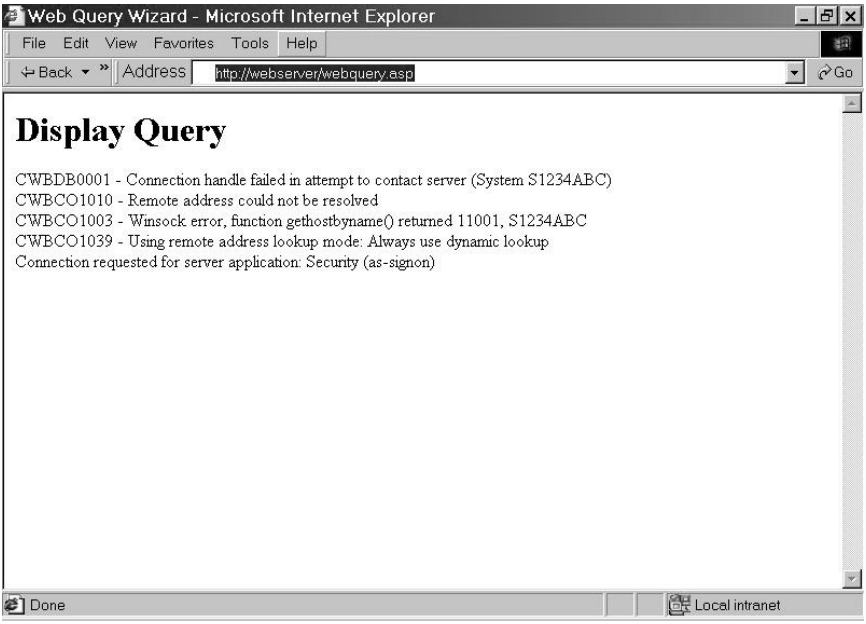


Abbildung 7.18: Alle vorhandenen Felder werden in einem neuen Browserfenster angezeigt.

Wenn keine Fehler auftreten, werden die ausgewählten Felddaten aus der definierten Datei nur für die Datensätze angezeigt, die in der WHERE-Klausel definiert werden.

Ein Beispiel für die fertige Ausgabe für WEBQUERY.ASP wird in Abbildung 7.19 gezeigt.

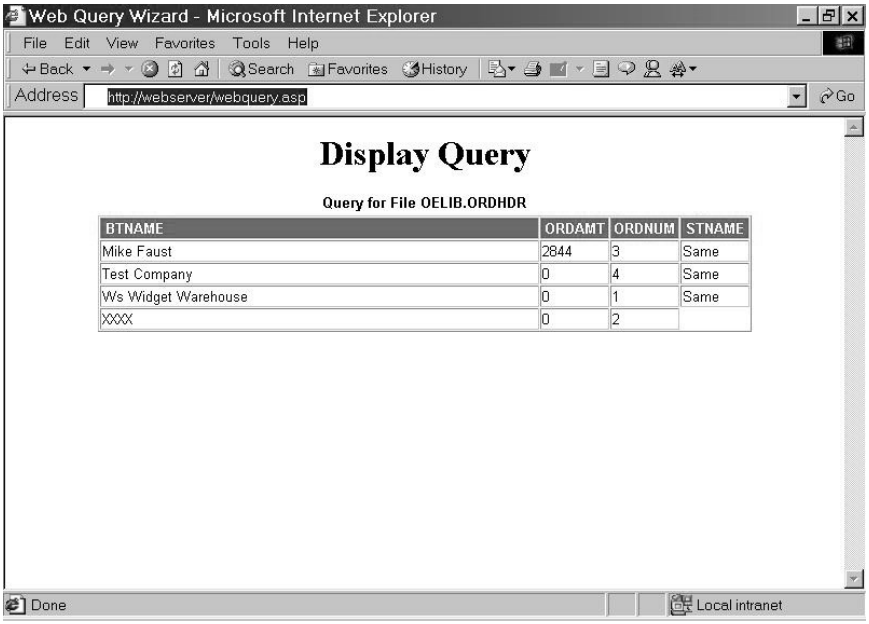


Abbildung 7.19: Das endgültige Ergebnis unserer Webabfrage zeigt ausgewählte Daten in einem Webbrowser an.

Dieses Beispiel bietet Ihnen ein nützliches Abfragetool, das Daten aus einer einzelnen Datei in Ihrer DB2/400-Datenbank abrufen kann. Alle hier gezeigten Beispiele vermitteln eine ausführliche Vorstellung davon, wie einfach es ist, Daten aus Ihrer DB2/400-Datenbank in eine Active Server Page zu integrieren. Sie haben eventuell bemerkt, dass wir die gleiche Technik wieder und wieder verwendet haben. Das ist der größte Vorteil der Verwendung von ASP und OLE DB: Sie können eine komplette Webseite mit OLE-DB-Zu-

griff auf der 400 erstellen, ohne dazu viel Code programmieren zu müssen. Ein weiterer Punkt, an den Sie denken sollten, ist es, dass Sie keine spezielle Druckfunktion in Ihre Anwendung integrieren müssen, weil der Benutzer diese Informationen in einem Webbrowser anzeigt. Sobald Sie die Daten generiert haben, kann der Benutzer einfach die Druckfunktion des Browsers verwenden. Auf jeden Fall erlauben es Ihnen Active Server Pages, Ihre Daten im Web in Echtzeit zu hosten, und das mit dem geringst möglichen Aufwand!

8

Internet-e-Mail von der iSeries

Zu den größten Vorteilen des Internets gehört die Tatsache, dass unterschiedliche Plattformen über Tools – wie Telnet, FTP und Webbrowser – miteinander kommunizieren können. Es ist wirklich erstaunlich, dass ein Computer, auf dem Linux ausgeführt wird, auf eine Webseite eines Windows-Server zugreifen kann, auf dem IIS ausgeführt wird, und von dem Daten von einer iSeries oder AS/400 abgerufen werden können. Ein weiteres Puzzleteil ist Internet-e-Mail. OS/400 hat die Fähigkeit, einfache Nachrichten oder sogar Objekte wie physische Dateien und Dokumente in freigegebenen Ordnern an beliebige Internet-e-Mail-Adressen zu senden. In diesem Kapitel befassen wir uns damit, wie Sie Ihre 400 konfigurieren müssen, um Nachrichten ins Internet zu senden. Zusätzlich erfahren Sie, wie eine Anwendung erstellt werden kann, um eine gespoolte Datei von der 400 ins Internet als Textdatei im Microsoft Rich-Text-Format (RTF) zu senden – d.h.: in einem von Microsoft Word lesbaren Format oder sogar als HTML-Dokument. Dieses Tool wird auf Ihrem System für die Berichtverteilung und Archivverwendung von unschätzbarem Wert sein.

Konfiguration

Der Distributionsdienst von OS/400 hat die Fähigkeit, eine Nachricht zu einem Simple Mail Transfer Protocol (SMTP) e-Mail Server zu senden. SMTP ist das primäre Nachrichtenformat, das verwendet wird, um e-Mails über das Internet zu senden. Dabei sind die Voraussetzungen an die Konfiguration minimal; die so gewonnene Funktionalität rechtfertigt diesen Aufwand durchaus. Die erste und wichtigste Voraussetzung ist, dass Sie Ihre 400 so konfigurieren, dass sie über TCP/IP kommuniziert. Zum Glück besteht die Aussicht, dass Ihr System bereits TCP/IP für die Kommunikation mit Ihrem LAN verwendet, so dass Sie nichts unternehmen müssen. Eine weitere wichtige Voraussetzung ist, dass Ihr Exchange Server einen Internet Mail Server ausführen muss. Wenn Sie e-Mails aus dem Internet empfangen und ins Internet senden können, ist das bereits der Fall. Nun müssen Sie Ihre AS/400 dazu bringen, mit Ihrem Exchange Server zu interagieren. Wir müssen die Distributionsdienste von OS/400 so einstellen, dass Ihre 400 e-Mails versenden kann. Im ersten Schritt muss hierfür ein Host-Tabelleneintrag für Ihren Exchange Server erstellt werden, wobei der Befehl Add TCP/IP Host Table Entry (ADDTCPHTE) verwendet wird:

```
ADDTCPHTE INTNETADR('xxx.xxx.xxx.xxx') HOSTNAME(Mailserver-Name).
```

Die x-Zeichen im ersten Parameter sollten mit der IP-Adresse Ihres Exchange Servers ersetzt werden. Der zweite Parameter gibt den Servernamen an, der dieser IP-Adresse zugewiesen wurde.

Als Nächstes muss der OS/400 SMTP-Server so konfiguriert werden, dass er auf den Mailserver verweist, indem der Befehl Change SMTP Attribute (CHGSMTPA) wie folgt eingegeben wird:

```
CHGSMTPA MAILROUTER(Mailserver-Name) AUTOSTART(*YES).
```

Der erste Parameter sollte dem Servernamen entsprechen, der im Befehl ADDTCPHTE angegeben wurde. Der zweite Parameter stellt sicher, dass der SMTP-Dienst startet, wenn die TCP/IP-Schnittstelle gestartet wird. Nun müssen Sie das Systemdistributionsverzeichnis mit einem Eintrag ergänzen, der als Route von der 400 zum Exchange Server genutzt werden wird. Hierzu geben Sie den Befehl Add Directory Entry (ADDDIRE) ein:

```
ADDDIRE USRID(INTERNET SMTPRTE) USRD('SMTP Mail Route')
        SYSNAME(INTERNET) PREFADR(NETUSRID *IBM ATCONTXT).
```

Der erste Parameter dieses Befehls gibt den Namen an, den wir diesem Verzeichniseintrag zuweisen. Der weitere wichtige Parameter ist hier der Befehl Preferred Address (PREFADR), von dem festgelegt wird, welcher Typ „Adressierung" erfolgen soll. Diese Route informiert OS/400 darüber, was mit den Distributionsobjekten geschehen soll, die an die Internet-e-Mail-Adresse versendet werden sollen. Danach müssen die Distributionsdienste angewiesen werden, diese Route zu verwenden, wenn sie SMTP-Mail verwenden. Dies erfolgt mit Hilfe des Befehls Change Distribution Attributes (CHGDSTA):

```
CHGDSTA SMTPRTE(INTERNET SMTPRTE)
```

Sie können die Konfiguration testen, indem Sie den SMTP-Server mit dem Befehl Start TCP/IP Server (STRTCPSVR) starten:

```
STRTCPSVR SERVER(*SMTP)
```

Schließlich müssen wir die Distributionsdienste mit Hilfe des Befehls Start Subsystem (STRSBS) wie folgt starten:

```
STRSBS SBS(?)
```

Wir müssen außerdem sicherstellen, dass für das Benutzerprofil der Benutzer, die eventuell e-Mail-Nachrichten von der 400 versenden möchten, ein Verzeichniseintrag vorhanden ist. Verwenden Sie den in Abbildung 8.1 gezeigten Befehl Work with Directory Entry (WRKDIRE), um die Liste der Verzeichniseinträge anzuzeigen.

```
                          Work with Directory Entries

Type options, press Enter.
  1=Add      2=Change   4=Remove   5=Display details   6=Print details
  7=Rename   8=Assign different ID to description   9=Add another description

Opt  User ID   Address   Description
 _   QDFTOWN   QDFTOWN   Default Owner
 _   QDOC      QDOC      Internal Document Owner
 _   QIBM      S1038BGM  ibm support
 _   QLPAUTO   QLPAUTO   Licensed Program Automatic User
 _   QLPINSTL  QLPINSTL  Licensed Program Install
 _   QPGMR     S1038BGM  QPGMR
 _   QSECOFR   QSECOFR   Security Officer
 _   QSMTPDMY  QSMTPSYS  QSMTPQ
 _   QSYS      QSYS      Internal System User Profile
 _   QSYSOPR   QSYSOPR   System Operator
 _   QUMB      QUMB      ULTIMEDIA SYSTEM FACILITIES
 _   QUSER     QUSER     Default user for PC Support
                                                                      More...
F3=Exit      F5=Refresh   F9=Work with nicknames   F11=Sort by description
F12=Cancel   F13=Work with departments   F17=Position to   F24=More keys
```

Abbildung 8.1: Der Bildschirm Work with Directory Entries wird verwendet, um die Systemverzeichnisinformationen zu ändern.

Verwenden Sie in diesem Bildschirm Option 1, um einen neuen Verzeichniseintrag für das benötigte Benutzerprofil zu erstellen. Wenn die in Abbildung 8.2 gezeigte Add-Directory-Entry-Anzeige erscheint, geben Sie die entsprechenden Informationen für Benutzer-ID/Adresse, Systemname und Benutzerprofilfelder ein. Wie Sie sehen können, sind die Benutzer-ID und das Benutzerprofil normalerweise identisch, ebenso wie die Adresse und der Systemname, der dem Systemnamen Ihrer 400 entsprechen sollte.

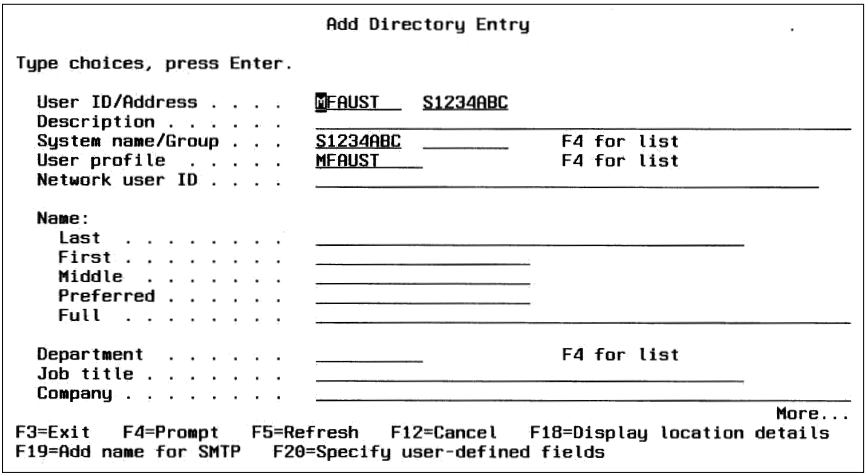


Abbildung 8.2: Dieser Bildschirm wird verwendet, um dem Systemverzeichnis einen neuen Eintrag hinzuzufügen.

Sie können auch die Namens- und Abteilungsdaten eingeben, obwohl diese nicht erforderlich sind. Dann drücken Sie F19, um einen SMTP-Namen für dieses Profil hinzuzufügen. Dieser wird als Absender- und Antwortadresse für alle Nachrichten verwendet, die vom angegebenen Benutzer versendet werden. Abbildung 8.3 zeigt Add Name für die SMTP-Anzeige.

Die SMTP-Benutzer-ID und SMTP-Domäne sollte dem Benutzer und der Domäne der Benutzer-e-Mail entsprechen. In diesem Beispiel lautet die fiktive e-Mail-Adresse *mfaust@ourserver. com*. Wenn Sie den SMTP-Namen festlegen, können Nachrichten sowie alle Nachrichtenversand-Fehlerbenachrichtigungen an Ihre tatsächliche e-Mail-Adresse versendet werden. Drücken Sie die Eingabetaste, um diese Daten zu speichern.

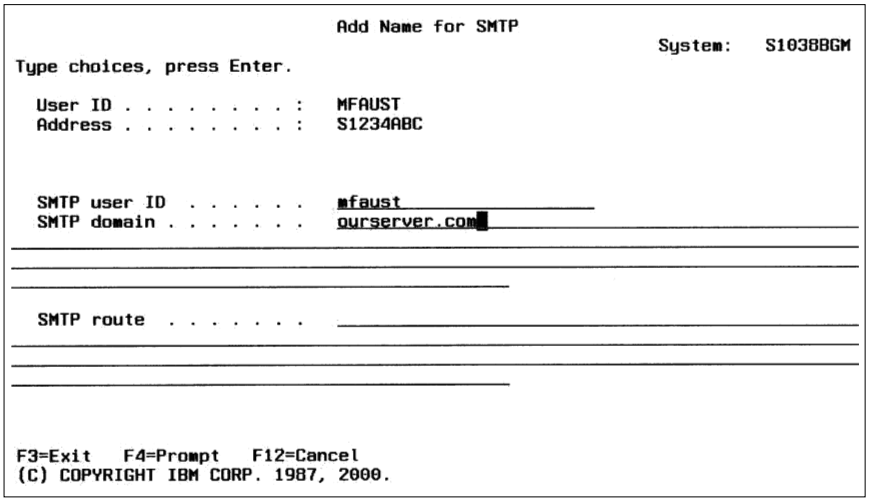


Abbildung 8.3: In diesen Bildschirm werden Internet (SMTP) e-Mail-Informationen für den angegebenen Benutzer eingegeben.

Nun sind wir so weit und können eine kurze Nachricht als Test versenden. Hierfür geben wir einen der Befehle ein, die mit OS/400 ausgeliefert werden, und die es uns ermöglichen, Nachrichten an eine e-Mail-Adresse zu senden. Der Befehl Send Distribution (SNDDST) kann Nachrichten und Objekte von der iSeries

oder der AS/400 senden. Geben Sie den Befehl SNDDST wie folgt ein:

```
SNDDST TYPE(*LMSG) TOINTNETADR(Ihre e-Mail-Adresse)
        DSTD('Testnachricht') LMSG('Das ist ein Test!!!').
```

Der erste Parameter definiert den Typ „Nachricht", der versendet wird. In diesem Fall wird der Wert *LMSG angegeben, um eine lange Nachricht an die definierte e-Mail-Adresse zu senden. Der Parameter TOINTNETADR definiert die e-Mail-Adresse für die zu sendende Nachricht. Der Parameter Distribution Description (DSTD) ist einfach ein kurzer Beschreibungstext, der die Nachricht beschreibt. Der Parameter Long Message (LMSG) schließlich wird für die e-Mail-Nachricht selbst verwendet. Unter der Voraussetzung, dass die Konfiguration korrekt ist, sollten Sie die e-Mail-Nachricht innerhalb von wenigen Minuten erhalten. Wenn das nicht der Fall ist, können Sie einige Dinge überprüfen. Erstens stellen Sie sicher, dass der SMTP-Dienst auf dem Mailserver ausgeführt wird, indem Sie den Befehl CWBPING Client Access eingeben. Dieser Befehl ist eine Version des Standard-PING-Befehls, der verwendet wird, um eine Verbindung mit einer angegebenen IP-Adresse zu testen. Der Befehl unterscheidet sich insofern von CWBPING, als dass er es Ihnen ermöglicht, **ping** für eine angegebene Portnummer auszuführen. Um den Standard-SMTP-Port zu testen, geben Sie diesen Befehl in eine Windows-Befehlszeile ein:

```
CWBPING (Mailserver IP-Adresse) /port:25
```

In diesem Beispiel wird die in Abbildung 8.1 gezeigte Nachricht ausgegeben, wenn der Maildienst fehlerfrei funktioniert. Wenn der SMTP-Dienst nicht ausgeführt wird, werden Fehler zurückgegeben.

```
C:\>cwbping 192.168.0.5 /port:25

IBM AS/400 Client Access Express for Windows
Version 4  Release 5  Level 0

(C) Copyright IBM Corporation and Others 1984, 1995.  All rights reserved.
U.S. Government Users Restricted Rights - Use, duplication or disclosure
  restricted by GSA ADP Schedule Contract with IBM Corp.
Licensed Materials - Property of IBM

To cancel the CWBPING request, press CTRL-C or CTRL-BREAK
I - Verifying connection to system 192.168.0.5...
I - Successfully connected to server application: #25
I - Connection verified to system 192.168.0.5

C:\>
```

Abbildung 8.4: Der Befehl CWBPING DOS wird verwendet, um den SMTP-Service zu testen.

Sie müssen auch sicherstellen, dass die Verteilerwarteschlangen aktiv sind. Dies geschieht, indem Sie das QSNADS-Subsystem mit folgendem OS/400-Befehl prüfen und falls erforderlich starten:

```
STRSBS SUBSYS(QSNADS)
```

Nachdem wir nun eine einfache Nachricht von der 400 in unsere Internet-e-Mail-Adresse gesendet haben, sehen wir uns an, welche weiteren Objekttypen gesendet werden können. Maßgeblich für den Versand von Nachrichten und Objekten von der 400 an das Internet-e-Mail-Konto ist der Befehl Send Distribution (SNDDST). Wir haben diesen Befehl bisher für das Senden von Testnachrichten von der 400 aus verwendet. In diesem Fall wird eine Nachricht vom Typ *LMSG gesendet. Eine vollständige Liste der Nachrichtentypen und Beschreibungen sehen Sie in Tabelle 8.1.

Tabelle 8.1: Gültige Codes für den Parameter TYPE des Befehls SNDDST	
Typ	**Beschreibung**
*MSG	Die Nachricht in Parameter MSG wird nur als Nachrichtentextkörper gesendet.
*DOC	Das Dokumentobjekt, das entweder im Parameter DOC oder DOCID definiert wird, wird als Anhang gesendet.
*FILE	Die Datenbankdatei, die für Parameter DOCFILE und DOCMBR definiert wurde, wird als Anhang gesendet.
*LMSG	Der Text für Parameter LMSG wird als Nachrichtentextkörper gesendet.
*DSTID	Der Maileintrag, der in Parameter DSTID definiert wird, wird gesendet.

Jeder dieser Typen dient einem bestimmten Zweck. Typ *MSG und *LMSG werden nur für den Versand einer Nachricht verwendet. Diese Funktionen sind z.B. nützlich, wenn eine Anwendung eine Benachrichtigung senden soll, sobald eine Aufgabe abgeschlossen wurde oder wenn ein Fehler aufgetreten ist. Folgendes CL-Beispiel sendet eine Nachricht an die angegebene e-Mail-Adresse:

```
IF COND(&GLERR='Y')  THEN(DO)

SNDDST TYPE(*MSG) MSG('Fehler in GL Batch vorhanden')
          TOINTNETADR(admin@ourmail.com)

ENDDO
```

In diesem Beispiel sendet die Anwendung diese Nachricht basierend auf einer erkannten Fehlerbedingung. Es können nicht nur einfache Nachrichten gesendet werden, um einen Benutzer über

eine festgelegte Bedingung zu informieren, sondern der Typ *FILE kann auch eine physische Datei an den Anwender senden. Ihre Anwendung kann Daten erstellen und dann an einen entsprechenden Benutzer senden, wenn der Typ *FILE angegeben wird. Dabei gilt es zu beachten, dass die Datei nicht konvertiert wird. Sie wird ohne Veränderung versendet. Der Dokumenttyp *DSDID kann verwendet werden, um eine eintreffende Nachricht an einen anderen Benutzer umzuleiten oder weiterzuleiten. Als Letztes folgt eine Erklärung des Nachrichtentyps *DOC. Dieser Typ ist sogar noch nützlicher als *FILE, weil mit *DOC ein Dokument in einem freigegebenen Ordner verknüpft wird. Das heißt, Sie können eine physische Datei in ein Dokument in einem freigegebenen Ordner konvertieren, bevor Sie diese versenden. Das kann sehr nützlich sein, wenn Ihre Anwendung Daten erstellt, die spezifisch über das Internet versendet werden sollen.

SMTP-Berichtverteilung

An diesem Punkt denken Sie eventuell: „Einen Augenblick mal, keiner der bereits beschriebenen Typen hatte irgendwas mit Berichten zu tun.“ Nun, der Prozess des Versands eines Berichts an eine Internet-e-Mail-Adresse erfordert die Kombination einiger weiterer Befehle. Als Erstes müssen Sie den Bericht in eine physische Datei speichern, indem Sie den Befehl Copy Spooled File (CPYSPLF) wie folgt eingeben:

```
CPYSPLF FILE(QPQXPRTF) TOFILE(QTEMP/SPOOLFILE)
        JOB(780198/MFAUST/QPADEV0054) SPLNBR(*LAST)
```

Als Nächstes müssen wir diese Datei in ein Dokument eines freigegebenen Ordners kopieren. Das erlaubt uns, die Datei vor dem Anhängen an eine e-Mail-Nachricht in das PC-Format umzuwandeln. Wir geben hierzu den Befehl Copy to PC Document (CPYTOPCD) wie folgt ein:

```
CPYTOPCD FROMFILE(QTEMP/SPOOLFILE) TOFLR(TEMP) TODOC(FILE.TXT)
         REPLACE(*YES)
```

In diesem Beispiel wird die Datei, die im letzten Schritt erstellt wurde, als PC-Dokument FILE.TXT im freigegebenen Ordner TEMP gespeichert. Zuletzt verwenden wir den Befehl Send Distribution (SNDDST) mit dem Nachrichtentyp *DOC wie folgt:

```
SNDDST TYPE(*DOC) TOINTNET((mfaust@ourserver.com)) DOC(FILE.TXT)
       FLR(TEMP)
```

Die Nachricht wird an die angegebene e-Mail-Adresse mit dem Anhang FILE.TXT gesendet. Ein Problem dieser Methode ist es, dass die meisten Berichte, die Sie auf Ihrer 400 generieren, für den Ausdruck aus den gängigen Textdatei-Anzeigeprogrammen nicht geeignet sind. Die Lösung für dieses Problem ist es, die Ausgabe vor dem Versand in ein Dateiformat zu konvertieren, das etwas „druckerfreundlicher" ist. Das folgende Beispiel erledigt nicht nur diese Aufgabe, sondern enthält auch einen neuen Befehl, mit dem der gesamte Prozess des Versands eines Berichts viel einfacher wird. Sie können einen Bericht an eine Internet-e-Mail-Adresse in einem der folgenden drei Formate versenden:

- *TEXT
- *RTF
- *HTML

Die erste Option sendet den Bericht als Textdatei. Die zweite Option sendet die Datei im Microsoft Rich-Text-Format, das mit Microsoft Word kompatibel ist. Dieses Format ist extrem „druckerfreundlich" und vermutlich das ideale Format für alle Berichte, die vom Client ausgedruckt werden. Das *HTML-Format schließlich sendet die Datei im Standard-Webbrowser-kompatiblen Hypertext Markup-Format. Dieses Format bietet die größtmögliche Client-Kompatibilität, da nahezu alle PC-Betriebssysteme eine HTML-Datei anzeigen können. Um diesen Befehl zu erstellen, prüfen wir als Erstes die CL-Source für den Befehl.

Abbildung 8.5 enthält den ersten Teil des CL-Programms EMR001CL.

```
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* PROGRAM NAME:EMR001CL                                                   */
/*                                                                         */
/* DESCRIPTION: E-MAIL REPORT IN TEXT, RTF OR HTML                         */
/*                                                                         */
/* Compile: CRTCLPGM PGM(QGPL/EMR001CL)                                    */
/*                                                                         */
/*-------------------------------------------------------------------------*/
             PGM        PARM(&FILENM &JOBN &SPLFN &EMAIL &REPDESC &FMT)
             DCL        VAR(&FILENM) TYPE(*CHAR) LEN(10) /* SPOOLED FILE +
                          NAME */
             DCL        VAR(&JOBN) TYPE(*CHAR) LEN(26) /* JOB NAME */
             DCL        VAR(&SPLFN) TYPE(*DEC) LEN(5 0) /* SPOOLED +
                          FILE # */
             DCL        VAR(&EMAIL) TYPE(*CHAR) LEN(40) /* EMAIL +
                          ADDRESS */
             DCL        VAR(&REPDESC) TYPE(*CHAR) LEN(50) /* REPORT +
                          DESCRIPTION */
             DCL        VAR(&FMT) TYPE(*CHAR) LEN(5) /* SEND FORMAT */
             DCL        VAR(&FEXT) TYPE(*CHAR) LEN(4) /* FILE +
                          EXTENSION */
             DCL        VAR(&SPL#) TYPE(*CHAR) LEN(5) /* SPOOLED +
                          FILE # */
             DCL        VAR(&FNAME)  TYPE(*CHAR) LEN(12)
             DCL        VAR(&JOBID)  TYPE(*CHAR) LEN(10)
             DCL        VAR(&USERID) TYPE(*CHAR) LEN(10)
             DCL        VAR(&JOBNBR)  TYPE(*CHAR) LEN(6)

             MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(ERRTRP))

/* CONVERT  SPLFN  VALUES  INTO  TEXT  VALUES */

             IF         COND(&SPLFN = -1) THEN(CHGVAR +
                          VAR(&SPL#) VALUE('*LAST'))
             ELSE       CMD(IF COND(&SPLFN = 0) THEN(CHGVAR VAR(&SPL#) +
                          VALUE('*ONLY')))
             ELSE       CMD(CHGVAR VAR(&SPL#) VALUE(&SPLFN))
```

Abbildung 8.5: Dieser Abschnitt des Programms von EMR001C ruft die Informationen zur gewünschten gespoolten Datei ab (Teil 1 von 2).

```
/* RETRIEVE JOB INFORMATION                                              */

             IF        COND(&JOBN = '*') THEN(RTVJOBA JOB(&JOBID) +
                         USER(&USERID) NBR(&JOBNBR))
             ELSE      CMD(DO)
             CHGVAR    VAR(&JOBID)  VALUE(%SST(&JOBN 1 10))
             CHGVAR    VAR(&USERID) VALUE(%SST(&JOBN 11 10))
             CHGVAR    VAR(&JOBNBR) VALUE(%SST(&JOBN 21 6))
             ENDDO

/* CONVERT FMT SPECIAL VALUES INTO FILE EXTENSIONS                       */

             IF        COND(&FMT = '*TEXT') THEN(CHGVAR VAR(&FEXT) +
                         VALUE('.TXT'))

             IF        COND(&FMT = '*RTF ') THEN(CHGVAR VAR(&FEXT) +
                         VALUE('.RTF'))

             IF        COND(&FMT = '*HTML') THEN(CHGVAR VAR(&FEXT) +
                         VALUE('.HTM'))
```

Abbildung 8.5: Dieser Abschnitt des Programms von EMR001C ruft die Informationen zur gewünschten gespoolten Datei ab (Teil 2 von 2).

Der erste Teil dieses CL-Programms übersetzt die Werte, die vom Befehl EMLRPT bereitgestellt werden, in Werte für die interne Verwendung durch das Programm. Die Jobinformationen z.B. werden in Job-ID, Benutzer-ID und Jobnummernfelder aufgespaltet. Ebenso wird die Dateierweiterung für die beendete Datei auf Grundlage des angeforderten Formats bestimmt.

Abbildung 8.6 zeigt den Rest des Codes für EMR001C.

```
/* Bericht an E-Mail-Adresse senden                                    */

               CHKOBJ      OBJ(QTEMP/SPLFIN) OBJTYPE(*FILE)
               MONMSG      MSGID(CPF9801) EXEC(CRTPF FILE(QTEMP/SPLFIN) +
                             RCDLEN(202) SIZE(*NOMAX))
               CHKOBJ      OBJ(QTEMP/RPTOUT) OBJTYPE(*FILE)
               MONMSG      MSGID(CPF9801) EXEC(CRTPF FILE(QTEMP/RPTOUT) +
                             RCDLEN(512) SIZE(*NOMAX))

               CPYSPLF     FILE(&FILENM) TOFILE(QTEMP/SPLFIN) +
                             JOB(&JOBNBR/&USERID/&JOBID) SPLNBR(&SPL#) +
                             MBROPT(*REPLACE) CTLCHAR(*PRTCTL)
               IF          COND(&FMT = '*RTF') THEN(DO)
               CLRPFM      FILE(RPTOUT)
               CALL        PGM(CVTRPTRTF)
               ENDDO

               IF          COND(&FMT = '*HTML') THEN(DO)
               CLRPFM      FILE(RPTOUT)
               CALL        PGM(CVTRPTHTML)
               ENDDO

               IF          COND(&FMT = '*TEXT') THEN(DO)
               CPYSPLF     FILE(&FILENM) TOFILE(QTEMP/RPTOUT) +
                             JOB(&JOBNBR/&USERID/&JOBID) SPLNBR(&SPL#) +
                             MBROPT(*REPLACE)
               ENDDO

               CHGVAR      VAR(&FNAME) VALUE(%SST(&FILENM 1 8) *TCAT &FEXT)
               CPYTOPCD    FROMFILE(QTEMP/RPTOUT) TOFLR('rpttemp') +
                             TODOC(&FNAME) REPLACE(*YES)
               CHGDOCD     DOC(&FNAME) FLR('rpttemp') DOCD(&REPDESC)
               SNDDST      TYPE(*DOC) TOINTNET((&EMAIL)) DSTD(&REPDESC) +
                             LONGMSG(&REPDESC) DOC(&FNAME) FLR('rpttemp') +
                             SUBJECT(&REPDESC)
               DLTDLO      DLO(&FNAME) FLR('rpttemp')

               RETURN
```

Abbildung 8.6: Der Abschnitt des CL-Programms, der eine gespoolte Datei zu einer Internet-Mailadresse sendet (Teil 1 von 2).

```
/* Fehler-Trapping     */

ERRTRP:         CALL        PGM(QMHRSNEM) PARM('    ' X'00000000')
                MONMSG      MSGID(CPF0000)

                SNDPGMMSG   MSGID(CPF9898) MSGF(QCPFMSG) MSGDTA('ERROR: +
                              SEE PERVIOUS MESSAGE IN THE JOBLOG FOR +
                              MORE INFORMATION') MSGTYPE(*ESCAPE)

                ENDPGM
```

Abbildung 8.6: Der Abschnitt des CL-Programms, der eine gespoolte Datei zu einer Internet-Mailadresse sendet (Teil 2 von 2).

Dieser Teil des Programms kopiert die gespoolte Datei in eine physische Datei, führt erforderliche Konvertierungen aus, kopiert die Datei in ein Dokument in einem freigegebenen Ordner und versendet sie mit dem Befehl SNDDST. Zusätzlich zu den zuvor beschriebenen drei Schritten, die erforderlich sind, um das Berichtsdokument zu versenden, konvertiert dieses Beispiel auch die Datei entweder in das RTF- oder in das HTML-Format und ändert die Dokumentbeschreibung für das PC-Dokument so ab, dass sie als Nachrichtenbetreffzeile verwendet werden kann. Dieses Programm wird mit dem Befehl Create CL Program erstellt:

```
RTCLPGM PGM(QGPL/EMR001CL)
```

Als Nächstes erstellen wir das ILE RPG-Programm, von dem die Ausgabe in eine RTF-Datei konvertiert wird. Bei dem in Abbildung 8.5 gezeigten Programm wird Ihnen auffallen, dass wir für den Befehl CPYSPLF den Control Character Parameter *PRTCTL für alle „nicht-textbasierten“ Formate setzen. Dieses Control Character Format fügt ein Space-Before-Feld (Leerraum vor) und ein Skip-Before-

Feld (Zeilenvorschub vor) vor dem eigentlichen Zeilentext ein. Die Konvertierungsprogramme verwenden diese Felder, um festzustellen, wie viele Seiten und Zeilen innerhalb Ihres Berichts vorhanden sind. Abbildung 8.7 zeigt die D-Spezifikationen aus der Source für das ILE RPG-Programm CVTRPTRTF.

```
 ************************************************************
 * Kompilierung:
 *              CRTBNDRPG(xxx/CVTRPTRTF) SRCFILE(xxx/QRPGLESRC) +
 *                 SRCMBR(xxx/QRPGLESRC)
 ************************************************************
FSPLFIN    IF   E             DISK    RECNO(RECN) RENAME(SPLFIN:SPLRCD)
FRPTOUT    O    F   198       DISK
 *
D RtfLine         S             198
D RTFHdrLn        S               1    DIM(100)
D RTFHdr          S             100    DIM(30) CTDATA PERRCD(1)
D LineLen         S               3  0
D RECN            S               4  0
D Lnt             S               3  0
D NotFirst        S                    LIKE(*IN01)
D CYr             S               4
D CMt             S               2
D CDy             S               2
 *
D PageCnt         S               4  0
D EOFSPLF         S                    LIKE(*IN01)
DCTime            DS              6
D CHr                     1       2
D CMn                     3       4
 *
DRecd             DS
DSpaceB                   1       3
DSkipB                    4       4
DRptLine                  5     202
```

Abbildung 8.7: Dieser Abschnitt des Programms definiert die Felder, die für die Konvertierung eines Berichts in RTF verwendet werden (Teil 1 von 2).

```
 *
DNewLine          C                   CONST('\line')
DNewPage          C                   CONST('\page')
DRptEnd           C                   CONST('\par}}')
D PgLn            S               5  0
DRptLn            S             198   DIM(100)
 *
ISPLRCD
I              SPLFIN                      SplInRcd
```

Abbildung 8.7: Dieser Abschnitt des Programms definiert die Felder, die für die Konvertierung eines Berichts in RTF verwendet werden (Teil 2 von 2).

Diese Definitionsspezifikationen enthalten Konstanten, die definiert werden, um das einfache Einfügen von Zeilen, Absätzen oder Seitenumbrüchen innerhalb der RTF-Datei zu ermöglichen. Abbildung 8.8 zeigt den zweiten Bereich dieses Programms.

```
 *
 */\/\/\/
```

```
 * Erste Ausgabe des Header-Abschnitts

 *
C                    Do        30            X                 3 0
 *
 * Nur Querformat wenn 132 oder 198
C                    If        LineLen<=80 And X=21
C                    Eval      RTFHdr(X)='\pszl                        '
C                    EndIf
 *
 * Datumsinformationen für Bericht aktualisieren
C                    If X=17
C                    MoveA     RTFHdr(X)     RTFHdrLn
C                    MoveA     CYr           RTFHdrLn(13)
C                    MoveA     CMt           RTFHdrLn(20)
C                    MoveA     CDy           RTFHdrLn(25)
C                    MoveA     CHr           RTFHdrLn(30)
C                    MoveA     CMn           RTFHdrLn(36)
C                    MoveA     RTFHdrLn      RTFHdr(X)
C                    EndIf
 *
 * Horiz Font Scaling basierend auf Seitengröße ändern
C                    If X=30
C                    MoveA     RTFHdr(X)     RTFHdrLn
 * 80
C                    If        LineLen=80
C                    MoveA     '150'         RTFHdrLn(21)
C                    EndIf
 * 132
C                    If        LineLen=132
C                    MoveA     '118'         RTFHdrLn(21)
C                    EndIf
 * 198
C                    If        LineLen=198
C                    MoveA     '79 '         RTFHdrLn(21)
C                    EndIf
 *
C                    MoveA      RTFHdrLn RTFHdr(X)
C                    EndIf
 *
C                    Eval       RtfLine=RTFHdr(X)
C                    Except     Rout
C                    EndDo
C                    Eval       PgLn=1
```

Abbildung 8.8: Dieser Teil des Programms richtet die RTF-Seite ein (Teil 2 von 2).

Der erste Teil der C-Spezifikationen führt als Erstes eine Routine aus, um die gesamte Datei einzulesen und festzustellen, wie viele Seiten sich im Dokument befinden und wie lang die Zeilen sein sollen. Als Nächstes initialisiert das Programm einige Variablen, die vom Programm verwendet werden und gibt dann die RTF-Header-Informationen aus, von denen die Formatierung des RTF-Dokuments abhängt. Der nächste Bereich des Codes, der in Abbildung 8.9 gezeigt wird, konvertiert die Berichtszeilen selbst.

```
 *
 * Nun wird die Berichtsdatei eingelesen und die Zeilenausgabe erstellt
 *
C                   Eval       EOFSPLF=*Off
C                   DoW        EOFSPLF=*Off
C                   Read       SPLFIN                                 LR
C                   Eval       EOFSPLF=%EOF(SPLFIN)
C                   If         EOFSPLF=*Off
C                   Eval       Recd = SplInRcd
 *
 * Leerschritt vor Wert-Neue Seite
 *
C                   If         SpaceB*BLANKS
C                   Move       SpaceB        Spc               3 0
C                   If         NotFirst=*On
C                   If         Spc<PgLn
C                   If         PgLn>0
C                   Do         PgLn          Ix                3 0
C                   Eval       RtfLine=RptLn(Ix)
C                   Except     Rout
C                   Eval       RTFLine=NewLine
C                   Except     Rout
C                   EndDo
C                   Clear                    RptLn
C                   EndIf
C                   Eval       RtfLine=NewPage
C                   Except     Rout
C                   EndIf
```

Abbildung 8.9: Dieser Teil unseres Programms konvertiert die einzelnen Berichtszeilen in RTF-Format (Teil 1 von 2).

```
C                   Else
C                   Eval       NotFirst=*On
C                   EndIf
C                   Eval       PgLn=Spc
C                   For        Ix=1 To LineLen
C                   If         %Subst(RptLn(PgLn):Ix:1)=' '
C                   Eval       %Subst(RptLn(PgLn):Ix:1) =
C                              %Subst(RptLine:Ix:1)
C                   EndIF
C                   EndFor
C                   Else
C                   Eval       NotFirst=*On
 *
 * Skip Before Value-Zeilen überspringen
 *
C                   If         SkipB*BLANKS
C                   Move       SkipB         Skp               1 0
C                   Eval       PgLn=PgLn+Skp
C                   For        Ix=1 To LineLen
C                   If         %Subst(RptLn(PgLn):Ix:1)=' '
C                   Eval       %Subst(RptLn(PgLn):Ix:1) =
C                              %Subst(RptLine:Ix:1)
C                   EndIF
C                   EndFor
C                   EndIf
C                   EndIf
C                   EndIf
C                   EndDo
 *
C                   If         PgLn>1
C                   Do         PgLn          Ix                3 0
C                   Eval       RtfLine=RptLn(Ix)
C                   Except     Rout
C                   Eval       RTFLine=NewLine
C                   Except     Rout
C                   EndDo
C                   EndIf
 *
C                   Eval       RtfLine=RptEnd
C                   Except     Rout
C                   Movel      *On           *INLR
C                   Return
```

Abbildung 8.9: Dieser Teil unseres Programms konvertiert die einzelnen Berichtszeilen in RTF-Format (Teil 2 von 2).

Dieser Codeabschnitt speichert die Daten für jede Seite in einem Array, das für jede Zeile der ausgedruckten Seite ein Element enthält. Dies geschieht, um Überdrucken zu ermöglichen. Das wird häufig verwendet, um Fettdruck oder Unterstreichen in einem Bericht zu erstellen. Die Daten für jede Seite werden anschließend aus diesem Array ausgegeben. Wie bereits erwähnt, wird anfangs eine Routine ausgeführt, um die Anzahl der Seiten und die Zeilenlänge für jeden Bericht festzustellen. Diese Routine wird in Abbildung 8.10 gezeigt.

Die Routine liest die Datei ein, indem sie nach Space-Before-Werten sucht. Ein Space-Before-Wert geringer als die aktuelle Zeilennummer steht für einen Zeilenumbruch. Diese Routine stellt auch die Länge der längsten Zeile fest und rundet diesen Wert auf die Standardseitenbreiten von 80, 132 oder 198 auf. Diese Daten werden später verwendet, um die RTF-Seite zu formatieren. Die Header-Informationen für das RTF werden in der Tabelle der Abbildung 8.11 gezeigt.

```
 *
 *
 *
 * Anzahl der Seiten basierend auf der Anzahl der SpaceB-Befehle zählen
 *
C     CntPages       Begsr
C                    Eval       PgLn=66
C                    Eval       PageCnt=0
C                    Eval       EOFSPLF=*Off
C                    DoW        EOFSPLF=*Off
C                    Read       SPLFIN
C                    Eval       EOFSPLF=%EOF(SPLFIN)
C                    If         EOFSPLF=*Off
```

Abbildung 8.10: Dieser Teil des Programms bestimmt, wie viele Berichtsseiten vorhanden sind (Teil 1 von 2).

```
C                   Eval       Recd = SplInRcd
C                   Eval       Lnt=%Len(%Trim(RptLine))
C                   If         Lnt>LineLen
C                   Eval       LineLen=Lnt
C                   EndIf
*
* Leerraum vor Wert–Neue Seite
*
C                   If         SpaceB*BLANKS
C                   Move       SpaceB   Spc
C                   If         Spc<=PgLn
C                   Eval       PageCnt=PageCnt+1
C                   Eval       PgLn=Spc
C                   EndIf
C                   Else
C                   Move       SkipB    Skp
C                   Eval       PgLn=PgLn+Skp
C                   EndIf
C                   EndIf
C                   EndDo
C                   Select
C                   When       LineLen<=80
C                   Eval       LineLen=80
C                   When       LineLen>80 And LineLen<=132
C                   Eval       LineLen=132
C                   When       LineLen<132 And LineLen<=198
C                   Eval       LineLen=198
C                   EndSl
* Datensatzposition zurücksetzen
C                   Eval       RECN=1
C     RECN          Setll      SPLFIN
C                   EndSr
```

Abbildung 8.10: Dieser Teil des Programms bestimmt, wie viele Berichtsseiten vorhanden sind (Teil 2 von 2).

```
     */\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
     ORPTOUT    E            Rout
     O                       RtfLine            198
** RTF Header Data (Inserted into the top of each Report
{\rtf1\ansi\ansicpg1252\uc1
\deff1\deflang1033\deflangfe1033{\fonttbl{\f0\froman
\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}{\f1\fswiss
\fcharset0\fprq2{\*\panose 20b0604020202020204}Arial;}{\f6\fmodern\fcharset0
\fprq1{\*\panose 00000000000000000000}Courier;}{\f83\froman\fcharset238\fprq2
Times New Roman CE;}{\f84\froman\fcharset204\fprq2 Times New Roman Cyr;}{\f86
\froman\fcharset161\fprq2 Times New Roman Greek;}{\f87\froman\fcharset162\fprq2
Times New Roman Tur;}{\f88\froman\fcharset186\fprq2 Times New Roman Baltic;}{\f89
\fswiss\fcharset238\fprq2 Arial CE;}{\f90\fswiss\fcharset204\fprq2 Arial Cyr;}{\f92
\fswiss\fcharset161\fprq2 Arial Greek;}{\f93\fswiss\fcharset162\fprq2 Arial Tur;}
{\f94\fswiss\fcharset186\fprq2 Arial Baltic;}}{\colortbl;\red0\green0\blue0;\red0
\green0\blue255;\red0\green255\blue255;\red0\green255\blue0;\red255\green0\blue255;
\red255\green0\blue0;\red255\green255\blue0;\red255\green255\blue255;\red0\green0
\blue128;\red0\green128\blue128;\red0\green128\blue0;\red128\green0\blue128;\red128
\green0\blue0;\red128\green128\blue0;\red128\green128\blue128;\red192\green192
\blue192;}{\stylesheet{\widctlpar\adjustright \f1\cgrid \snext0 Normal;}{\*\cs10
\additive Default Paragraph Font;}}{\info{\author CVTRPTRTF}{\operator CVTRPTRTF}
{\creatim\yr2000\mo01\dy27\hr09\min27}{\revtim\yr2000\mo01\dy27\hr09\min36}{\version1}
{\edmins1}{\nofpages1}{\nofwords0}{\nofchars0}{\*\company Midrange Computing}
{\nofcharsws0}{\vern113}}\margl259\margr245\margt245\margb230
\widowctrl\ftnbj\aenddoc\hyphcaps0\formshade\viewkind1\viewscale100\pgbrdrhead\pgbr
drfoot\fet0\sectd
\lndscpsxn\paperw15840\paperh12240
\psz1\linex0\endnhere\sectdefaultcl {\*\pnseclvl1\pnucrm\pnstart1
\pnindent720\pnhang{\pntxta
.}}{\*\pnseclvl2\pnucltr\pnstart1\pnindent720\pnhang{\pntxta .}
}{\*\pnseclvl3\pndec\pnstart1\pnindent720\pnhang{\pntxta
.}}{\*\pnseclvl4\pnlcltr\pnstart1
\pnindent720\pnhang{\pntxta
)}}{\*\pnseclvl5\pndec\pnstart1\pnindent720\pnhang{\pntxtb (}
{\pntxta )}}{\*\pnseclvl6\pnlcltr\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta
)}}
{\*\pnseclvl7\pnlcrm\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta
)}}{\*\pnseclvl8
\pnlcltr\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta
)}}{\*\pnseclvl9\pnlcrm\pnstart1
\pnindent720\pnhang{\pntxtb (}{\pntxta )}}\pard\plain \widctlpar\adjustright
\f1\cgrid
{\f6\fs16\charscalex79
```

Abbildung 8.11: Dieser RTF-Code wird verwendet, um die Seitenformatierung zu definieren.

Dieser Code sieht kryptisch aus, ähnelt jedoch HTML. Die gezeigten Informationen werden ganz oben in die einzelnen RTF-Dateien eingefügt und definieren Seitengröße und Ausrichtung sowie die Fonts, die für das Dokument verwendet werden sollen. Die Ausgabe, die von diesem Programm erstellt wird, wird anschließend in ein PC-Dokument konvertiert, wobei der Befehl CPYTOPCD verwendet wird, den wir bereits kennen.

Das Programm CVTRPTHTML, das von der angegebenen Webseite heruntergeladen werden kann, führt eine ähnliche Konvertierung in das HTML-Format aus. Die grundsätzliche Logik dieses Programms ähnelt der Logik des Befehls CVTRPTRTF. Der wichtigste Unterschied ist die Tatsache, dass das Programm HTML-Tags und Hyperlinks erstellt, die es einem Benutzer ermöglichen, zur ersten, zur vorherigen, zur nächsten oder zur letzten Seite des Berichts zu wechseln. Die Ausgabe dieses Programms kann in einem Standard-Webbrowser angezeigt werden. Das Programm bestimmt auch als Erstes die Anzahl der Seiten und die maximale Zeilenlänge. Als Nächstes werden HTML-Header ausgegeben. Diese Header bestimmen die Breite für die HTML-Tabelle, die erstellt wird, um die Berichtsausgabe aufzunehmen. Der Font wird auf Courier New gesetzt, weil die Weite dieses Fonts nicht proportional ist (die Verwendung eines proportionalen Fonts kann dazu führen, dass Spaltenüberschriften und Spaltendaten nicht richtig ausgerichtet sind).

Die Berichtsausgabe wird in eine HTML-Tabelle eingefügt, da dies eine einfache Methode ist, um sicherzustellen, dass die Berichtszeilen bei der Anzeige im Browser nicht automatisch umgebrochen werden. Bevor Sie diese beiden ILE RPG-Programme kompilieren, müssen Sie die physische Datei SPLFIN kompilieren. Dies ist die extern definierte Datei, die vom Befehl CPYSPLF verwendet

wird. Um diese Datei zu erstellen, geben wir den folgenden Befehl CPYSPLF ein:

```
CPYSPLF FILE(QPQXPRTF) TOFILE(LIB/SPLFIN)
JOB(780198/MFAUST/QPADEV0054)
        SPLNBR(*LAST)
```

Ersetzen Sie die Parameter FILE und JOB mit gültigen Parametern Ihres Systems. Diese Informationen erhalten Sie in der Anzeige Work with Spooled Files (WRKSPLF), wenn Sie Option 1 für die gewünschte Datei wählen. Sobald die Datei erstellt wurde, können Sie die beiden Programme erstellen, indem Sie folgenden Befehl Create Bound RPG Program (CRTBNDRPG) eingeben:

```
CRTBNDRPG(lib/CVTRPTRTF) SRCFILE(lib/QRPGLESRC) SRCMBR(lib/QRPGLESRC)

CRTBNDRPG(lib/CVTRPTHTML) SRCFILE(lib/QRPGLESRC) SRCMBR(lib/QRPGLESRC)
```

Nun müssen wir den Befehl erstellen, mit dem alle Teile wieder zusammengefügt werden. Abbildung 8.12 enthält die Befehls-Source für den Befehl Send Report to e-Mail.

```
/* Bericht im Text-, RTF- oder HTML-Format via E-Mail versenden */
/*                                                              */
/* Kompilierung:  CRTCMD CMD(*LIBL/SNDRPTEML)                   */
/*                       PGM(*LIBL/EMR001CL)                    */
/*                                                              */

                  CMD         PROMPT('Send Report via E-Mail')

                  PARM        KWD(FILE) TYPE(*NAME) MIN(1) PROMPT('Report +
                                (spool file)')

                  PARM        KWD(JOB) TYPE(JOB) DFT(*) SNGVAL((*)) +
                               PROMPT('Job Name')
```

Abbildung 8.12: Der Befehl SNDRPTEML sendet eine gespoolte Datei zu einer angegebenen e-Mail-Adresse (Teil 1 von 2).

```
                PARM        KWD(SPLNBR) TYPE(*DEC) LEN(4) DFT(*LAST) +
                              SPCVAL((*ONLY 0) (*LAST -1)) +
                              PROMPT('Spool File #')

                PARM        KWD(EMAIL) TYPE(*CHAR) LEN(40) MIN(1) +
                              PROMPT('E-Mail Address')

                PARM        KWD(REPDESC) TYPE(*CHAR) LEN(50) +
                              PROMPT('Message Subject')

                PARM        KWD(FMT) TYPE(*CHAR) LEN(5) RSTD(*YES) +
                              DFT(*RTF) VALUES(*TEXT *RTF *HTML) +
                              PROMPT('Transfer to Format')

JOB:            QUAL        TYPE(*NAME)
                QUAL        TYPE(*NAME) PROMPT('User ID')
                QUAL        TYPE(*CHAR) LEN(6) RANGE(000000 999999) +
                              PROMPT('Job Number')
```

Abbildung 8.12: Der Befehl SNDRPTEML sendet eine gespoolte Datei zu einer angegebenen e-Mail-Adresse (Teil 2 von 2).

Dieser Befehl hat sechs Parameter:

- **FILE**: Spezifiziert den gespoolten Dateinamen.
- **JOB**: Definiert den Job, von dem die gespoolte Datei generiert wurde, im Format Job #/User/Job-Name.
- **SPLNBR**: Identifiziert eine bestimmte Datei innerhalb des Jobs, wenn mehrere Dateien mit dem gleichen Namen innerhalb des gleichen Jobs vorhanden sind. Neben dem numerischen Wert wird die Datei, die zuletzt generiert wurde, mit dem Sonderwert *LAST identifiziert. *ONLY definiert, dass diese Datei nur einmal innerhalb des Jobs auftritt.
- **EMAIL**: Definiert die Internet-e-Mail-Adresse, an die dieser Bericht gesendet werden soll.

- **REPDESC**: Definiert den Nachrichtenbetreff für die e-Mail-Nachricht.
- **FMT**: Identifiziert das gewünschte Dateiformat für den Bericht. Die gültigen Sonderwerte für diesen Parameter lauten *TEXT für eine Textdatei, *RTF für eine Microsoft Rich-Text-Datei oder *HTML für eine Hypertext-Markup-Datei.

Erstellen Sie diesen Befehl mit Create Command (CRTCMD):

```
CRTCMD CMD(*LIBL/SNDRPTEML) PGM(*LIBL/EMR001CL)
```

Um den neu erstellten Befehl zu testen, geben Sie SNDRPTEML ein und drücken die Taste F4, um die Befehlsbedienerführung anzuzeigen. Es sollte der Bildschirm in Abbildung 8.13 angezeigt werden.

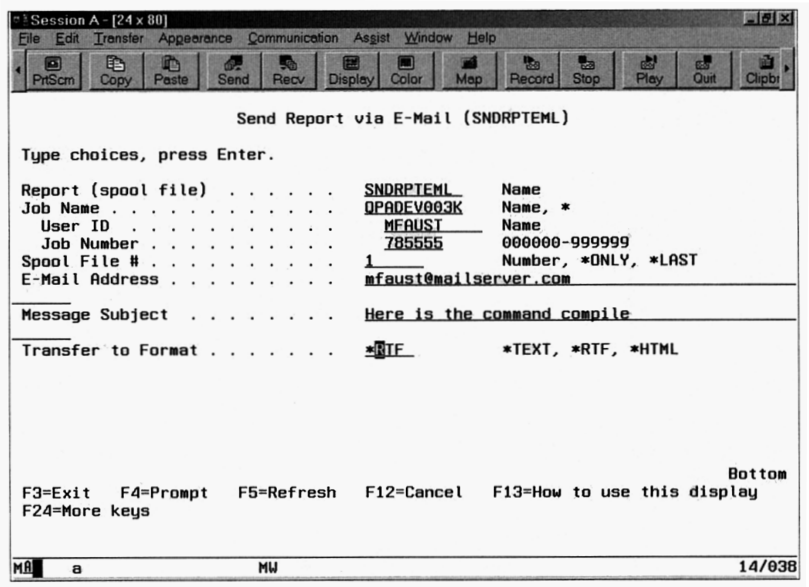


Abbildung 8.13: Bedienerführungsbildschirm für den Befehl SNDRPTEML

Dieses Beispiel sendet den Kompilierbefehl des Befehls CRTCMD an die angegebene e-Mail-Adresse. Die HTML- und RTF-Formate unterscheiden sich grundlegend voneinander. Die Abbildungen 8.14 und 8.15 enthalten eine Gegenüberstellung mit dem Compiler-Bericht. Zuerst (Abbildung 8.14) sehen Sie die RTF-Datei, die in Microsoft Word geöffnet wird, nachstehend (Abbildung 8.15) sehen Sie dann die HTML-Datei, die im Internet Explorer angezeigt wird.

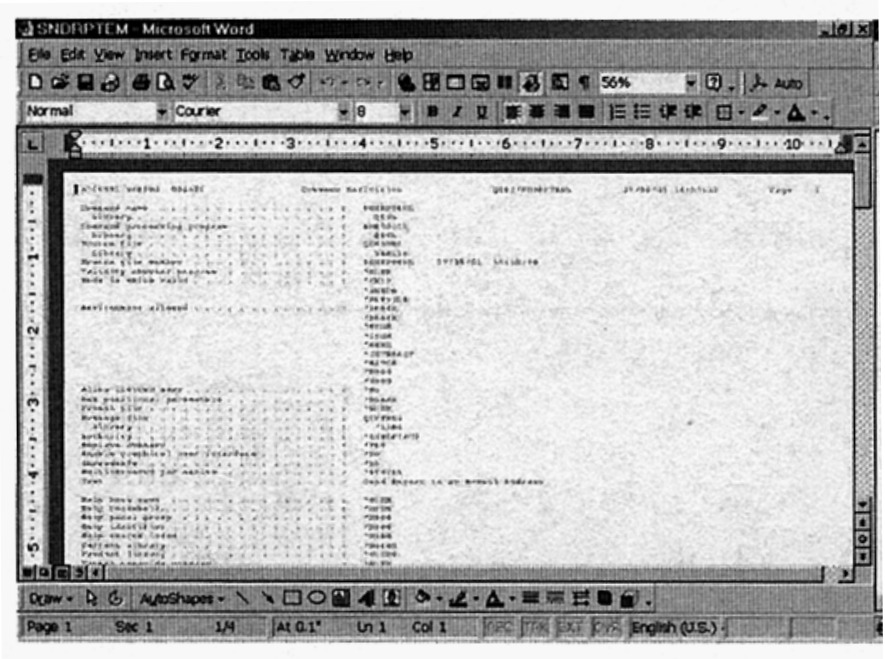


Abbildung 8.14: Der Befehl SNDRPTEML kann im RTF-Format senden.

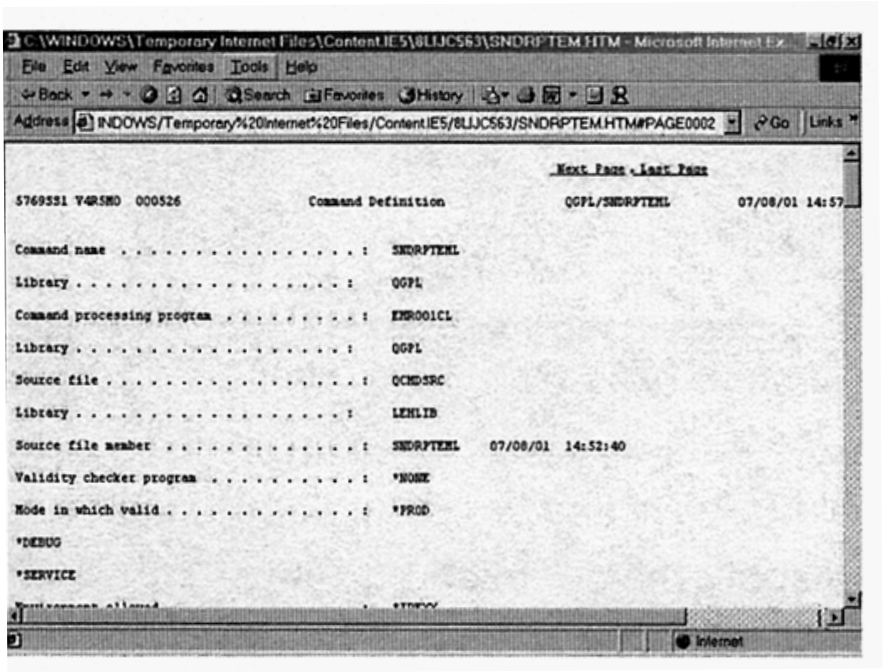


Abbildung 8.15: Der Befehl SNDRPTEML kann auch im HTML-Format senden.

Wie Sie sehen, sind diese beiden Versionen ähnlich, jedoch für unterschiedliche Anwendungen geeignet. Nachdem wir nun den Befehl erstellt haben, sehen wir uns an, welche weiteren Verwendungsmöglichkeiten sich bieten.

Erweiterung der SNDRPTEML-Funktionalität

Der Befehl SNDRPTEML genügt, um Ihre 400 mit umfangreicher Funktionalität zu erweitern. Wir können aber noch einiges tun, um diese Funktionalität zu vergrößern und benutzerfreundlicher zu machen. Ein Problem mit diesem Befehl besteht darin, dass Sie den Namen der gespoolten Datei und die Jobinformationen kennen müssen, bevor Sie den Befehl einsetzen. Idealerweise wäre es praktisch, wenn Sie auf diesen Befehl von der Anzeige Work with Spooled Files aus zugreifen könnten. Hierzu muss der Befehl Send Network Spooled File (SNDNETSPLF) mit unserem neu erstellten Befehl ergänzt werden. Dieser Befehl wird verwendet, wenn Option 1 – Send – im Bildschirm Work with Spooled File gewählt wird. Vergessen Sie dabei nicht, dass Sie, wenn Sie den Befehl SNDNETSPLF verwenden, nun die Art und Weise ändern müssen, in der Sie auf diesen Befehl zugreifen. Um den Befehl SNDNETSPLF zu überschreiben, erstellen Sie zuerst eine neue Bibliothek, die modifizierte Versionen der IBM-Befehle enthält.

Sie haben sicher bereits eine Bibliothek für diesen Zweck eingerichtet; wenn nicht, sollten Sie den folgenden Befehl Create Library (CRTLIB) eingeben:

```
CRTLIB LIB(CMDOVRD) TYPE(*PROD) TEXT('Command Override Library')
```

Nun modifizieren Sie den Standardsystembereich der Bibliotheksliste und fügen Ihre neue Bibliothek oberhalb von QSYS in der Bibliotheksliste ein. Verwenden Sie die Anzeige Work with System Value, um den Systemwert zu ändern, indem Sie WRKSYSVAL SYSVAL(QSYSLIBL) eingeben und Option 2 für diesen Systemwert wählen. Abbildung 8.16 zeigt den Bildschirm, der angezeigt wird.

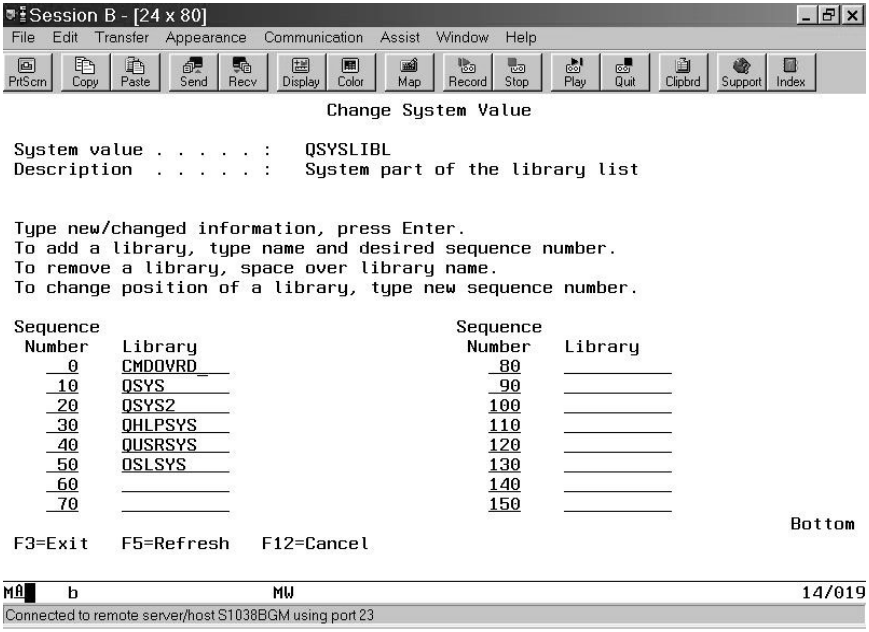


Abbildung 8.16: Verwenden Sie diesen Bildschirm, um eine neue Bibliothek ganz oben in die Systembibliotheksliste einzufügen.

Nun kopieren Sie den Befehl SNDRPTEML in die Bibliothek CMDOVRD und benennen sie in NDNETSPLF um. Hierzu verwenden Sie den Befehl Create Duplicate Object (CRTDUPOBJ):

```
CRTDUPOBJ OBJ(SNDRPTEML) FROMLIB(lib) OBJTYPE(*CMD)
                         TOLIB(CMDOVRD) NEWOBJ(SNDNETSPLF)
```

Jedes Mal, wenn ein Benutzer in der Anzeige Work with Spooled Files Option 1 wählt, erhält er Zugriff auf den Befehl Send Report to E-Mail. Eine Warnmeldung wird angezeigt, die Sie darüber in-

formiert, dass der Parameter TOUSRID nicht gültig ist, aber dies beeinträchtigt nicht die Funktionalität des Befehls.

Eine weitere Methode für die Verwendung des Befehls SNDRPTEML besteht darin, Berichte von der 400 mit Hilfe der Exchange Server Public Folder zu archivieren. Diese Ordner dienen als allgemeiner Speicherbereich innerhalb eines Exchange-Servers, auf den jeder autorisierte Benutzer zugreifen kann. Sie können den Befehl SNDRPTEML problemlos in Ihre täglich, wöchentlich oder monatlich ausgeführten Prozesse aufnehmen, um bestimme Berichte in einen öffentlichen Archivordner zu senden. Zusätzlich zur Archivierung gewähren Sie so mehreren Benutzern einfachen Zugriff auf diese Berichte, ohne den Bericht zu diesen Benutzern senden zu müssen. Wenn Sie die richtige Berechtigung aufweisen, können Sie von Microsoft Outlook aus einen neuen öffentlichen Ordner erstellen, indem Sie zu Public Folders wechseln. Anschließend wählen Sie zu All Public Folders und New Folder aus dem Submenü Folders im Menü File. Die Tastenkombination Control-Shift-E kann ebenfalls dafür verwendet werden. Sie müssen nur einen Ordnernamen angeben. Optional können Sie auch den Typ der Objekte angeben, die im Ordner gespeichert werden sollen, sowie einen Unterordner von All Folders, der den neuen Ordner aufnehmen kann.

Abbildung 8.17 zeigt das Fenster Create New Folder von Microsoft Outlook.

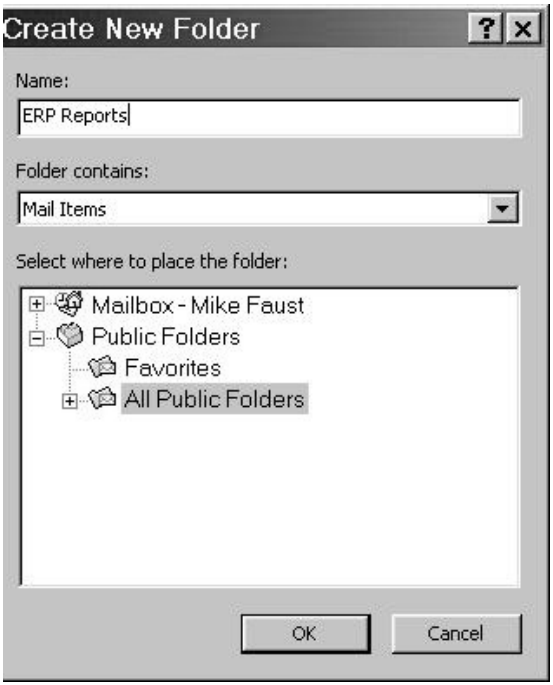


Abbildung 8.17: Verwenden Sie dieses Fenster, um einen neuen öffentlichen Exchange Folder zu erstellen.

Die SMTP-e-Mail-Adresse ist standardmäßig der Ordnernamen abzüglich der enthaltenen Leerstellen – gefolgt von @servername.com. Im Beispiel von Abbildung 8.12 lautet die SMTP-e-Mail-Adresse: *erpreports@servername.com*. Wenn auf Ihrem Exchange Server mehrere Ordner mit dem gleichen Namen vorhanden sind – z.B. wenn Ordner unterhalb der All-Folders-Gruppe für jede Abteilung Ihres Unternehmens vorgesehen sind und sich innerhalb der einzelnen Ord-

ner jeweils ein Ordner namens ERP Reports befinden würde – erhält der erste Ordner den Namen *erpreports@servername.com*. Der nächste Ordner erhält die Bezeichnung *erpreports1@servername.com* usw. Um den Namen zu ändern, öffnen Sie den Microsoft Exchange-Administrator und klicken auf den Ordnernamen unter Folders/Public Folders. Als Nächstes wählen Sie im Menü File die Option Properties und klicken auf die e-Mail-Adress-Registerkarte. Abbildung 8.18 zeigt dieses Fenster.

Abbildung 8.18: Das Dialogfeld Folder Properties wird verwendet, um eine Public Folder SMTP-Mail-Adresse zu ändern.

Um die Adresse zu ändern, klicken Sie auf die Schaltfläche Edit und modifizieren den Namensbereich der Adresse. Ihre Anwendung auf der 400 sendet anschließend Daten an diese Adresse anstatt zur e-Mail-Adresse eines bestimmten Benutzers. Diese Methode ist für die Distribution von Berichten ideal. Denn nun genügt es, eine Kopie eines Berichts zu einem Zielort zu senden, der für die jeweiligen Mitarbeiter zugänglich ist, anstatt die einzelnen Kopien eines Berichts an jede Person zu senden, die diese Informationen benötigt. Für Archivdateien ist diese Methode ebenfalls gut geeignet, weil z.B. jede Person, die einen Bericht vom letzten Juni sucht, nur den richtigen öffentlichen Ordner öffnen muss und auf Grundlage der gesendeten Daten eine Suche ausführen kann.

Noch eines sollte erwähnt werden, bevor dieses Kapitel abgeschlossen wird: Sie könnten Probleme mit Ihrem Exchange Server bekommen, der es Ihnen eventuell nicht erlaubt, Nachrichten an eine externe SMTP-Mail-Adresse zu senden. Das kann vorkommen, wenn Ihr Exchange Server so konfiguriert ist, dass e-Mail-Weiterleitung nicht zulässig ist. Damit soll häufig *Spamming* oder der unverlangte Versand von e-Mail-Nachrichten verhindert werden. Gehen Sie wie folgt vor, um dieses Problem zu lösen. Wechseln Sie im Exchange-Administrator unter Configuration für den entsprechenden Exchange-Standort zu Connections und zeigen Sie dann die SMTP-Eigenschaften mit Doppelklick auf Internet Mail Service an. Als Nächstes wählen Sie das Routing-Register und klicken auf die Schaltfläche Routing Restrictions.

Der in Abbildung 8.19 gezeigte Dialog wird dann angezeigt.

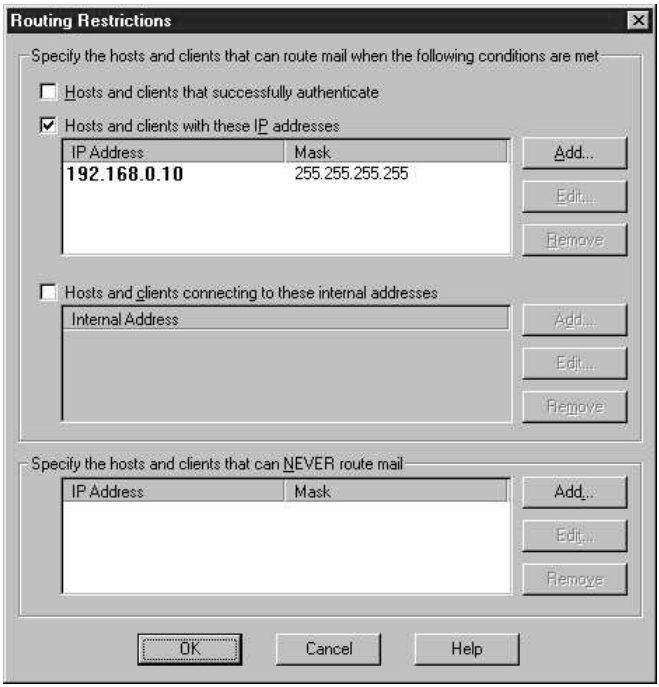


Abbildung 8.19: Der Dialog für die Routing-Einschränkungen wird für das SMTP-Nachrichten-Forwarding verwendet.

Das zweite Kontrollkästchen wird verwendet, um die Systeme zu definieren, die für die Weiterleitung von Nachrichten über den Exhanges Server zugelassen sind. In diesem Beispiel werden unter der Voraussetzung, dass die 400 die IP-Adresse 192.168.0.10 aufweist, Nachrichten weitergeleitet. Ein neuer Eintrag kann durch Klicken auf die Schaltfläche Add hinzugefügt werden. Geben Sie einfach die IP-Adresse Ihres Systems gemeinsam mit der Adresse

der zu verwendenden Maske ein. Diese Maske bezieht sich nicht auf Ihre IP-Netzwerk-Maske, sondern nur auf die Routing-Einschränkungen. In unserem Fall wird 255.255.255.255 verwendet, um anzugeben, dass nur diese Adresse Zugriff auf das SMTP-Nachrichten-Forwarding erhalten soll.

Wie wir in diesem Kapitel festgestellt haben, werden die Möglichkeit, e-Mail-Nachrichten von Ihrer 400 via SMTP zu senden, sowie die Art und Weise, in der Sie Informationen innerhalb Ihres Unternehmens verteilen, von Grund auf verändert. Wenn Sie Funktionalität für das Senden von Berichten über SMTP-Mail und Exchange Public Folder mit dem Befehl SNDRPTEML hinzufügen, können Sie bei der Berichtverteilung Zeit und Arbeit sparen. Und dabei können Sie sogar noch ein paar Bäume retten.

9

Internet für die 400

Bis vor kurzer Zeit waren die Möglichkeiten für die Verbindung mit einer AS/400 oder iSeries über das Internet noch begrenzt. Meist beinhaltet der Prozess die Verwendung des IBM Workstation Gateways und die Öffnung Ihrer 400 zum Internet hin. Die Sicherheitsrisiken dieser Vorgehensweise überwiegen jedoch bei weitem den Nutzen, der Ihnen das Hinzufügen dieser Funktionalität zu Ihrem System bietet. Vor kurzem wurden jedoch einige Produkte auf den Markt gebracht, die 5250-Konnektivität über einen Webbrowser bereitstellen, ohne dass die 400 im Internet gehostet werden muss. Das Internet ermöglicht nicht nur die Verbindung mit einer 400 über das Internet, sondern die Nutzung des Internets durch die 400 kann Ihre vorhandenen Green-Screen-Anwendungen mit umfangreicher Funktionalität erweitern.

Web-to-Host-Konzepte

Der Begriff „Web-to-Host“ umfasst eine breite Palette von Produkten, die sich mit dem Einsatz eines Hostsystems für das Internet befassen. Insbesondere sprechen wir hier von Programmen, mit denen die Ausführung einer Terminalemulationssitzung auf der 400 über einen Webbrowser möglich wird. Derzeit gibt es eine große Anzahl von Herstellern für Produkte für Web-to-Host-Konnektivität. Wie diese im einzelnen Web-to-Host-Konnektivität erzielen, variiert bei den einzelnen Produkten. Prinzipiell gibt es drei Grundkategorien von Produkten vom Typ „Web-to-Host“:

- Direktverbindung: Grundsätzlich handelt es sich hier um einen Java- oder ActiveX TN5250-Emulator, der in einer Webseite eingebettet ist. Die 400 muss über eine IP-Adresse verfügen, auf die über das Internet zugegriffen werden kann.
- Verbindung via Server: Ein Server agiert als „TN5250-Proxy“, der es Ihnen erlaubt, Ihre 400 hinter der Firewall auszuführen. Der Server stellt die Verbindungen zur 400 her.
- HTML-Transformationsprodukte: Konvertiert Daten vom 5250-Datenstrom in HTML, das an den Browser gesendet wird.

In diesem Kapitel werden Ihnen die einzelnen Typen sowie deren Vorteile und Nachteile vorgestellt. Eine Direktverbindungsoption kann am einfachsten eingerichtet werden, weil es sich prinzipiell um einen Telnet 5250-Emulator in einem Java- oder ActiveX-Control handelt. Dieses Objekt wird in eine Webseite eingefügt, auf die von einem beliebigen Webbrowser zugegriffen werden kann. Abbildung 9.1 enthält ein Diagramm zur Funktionsweise dieses Produkttyps.

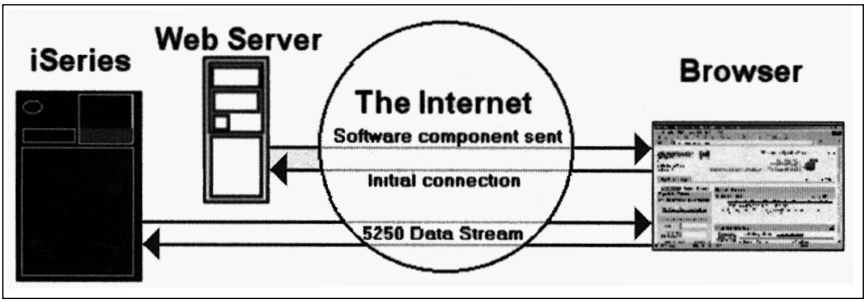


Abbildung 9.1: Dieser Typ „Web-to-Host-Anwendung“ erfordert, dass Ihre 400 eine direkte Webverbindung besitzt.

Wie dieses Bild zeigt, müssen für die direkte Web-Verbindungsmethode sowohl der Webserver, der die Web-to-Host-Webseite enthält, als auch die iSeries selbst direkt mit dem Internet verbunden sein. Der Browser-Client greift auf den Webserver zu, der eine Webseite sendet, die den vorkonfigurierten Telnet 5250-Client entweder als Java Applet oder ActiveX Control enthält. Wenn die Seite im Client-Browser angezeigt wird, wird eine neue Verbindung zwischen der 400 bzw. der iSeries über das Internet aufgebaut. Ab diesem Punkt findet die Kommunikation nur noch zwischen dem Browser-Client und der iSeries statt. Das bietet den Vorteil, dass nur wenig Konfigurationsarbeiten erforderlich sind. Sie können ohne großen Aufwand und ohne zusätzliche Dienste auf Ihrem Server die Verbindung von Remote-Anwendern mit Ihrer 400 erlauben. Das bedeutet auch, dass Web-to-Host-Software-Aktualisierungen nur auf dem Webserver ausgeführt werden und den Clients automatisch bei der Anmeldung bereitgestellt werden. Der offensichtliche Nachteil hiervon ist, dass die 400 selbst im Internet sein muss, was zusätzliche Sicherheitsrisiken bedeutet. Wenn wichtige Unternehmenssysteme auf der 400 installiert sind,

mit der sich der Web-to-Host verbindet, ist das möglicherweise nicht die richtige Methode für Sie. Dem gegenüber kann das eine geeignete Lösung für Sie sein, wenn Sicherheit für Sie aus irgendeinem Grund weniger wichtig ist.

Wenn Sie zu dem Schluss kommen, dass die Direktverbindungsmethode für Sie nicht geeignet ist, können Sie sich eventuell für eine Serverlösung entscheiden. In diesem Fall bietet der Server einen zusätzlichen Dienst, von dem die Kommunikation mit der iSeries abgewickelt wird. Abbildung 9.2 ist eine Übersicht über den Fluss eines Web-to-Host-Produkts.

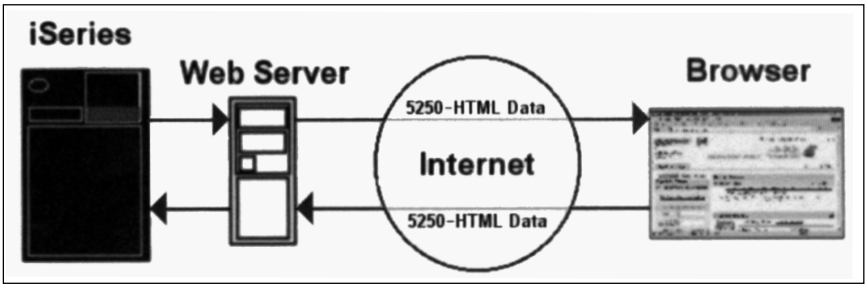


Abbildung 9.2: Dieser Web-to-Host-Typ greift auf den Host über einen Webserver zu.

In diesem Beispiel kommuniziert der Browser nur mit dem Webserver. Wie bei der vorherigen Methode sendet der Webserver als Erstes den Telnet-Client zum Browser. Der Webserver wickelt die gesamte Kommunikation mit dem iSeries-Host ab. Die Web-to-Host-Software auf dem Webserver agiert als *Telnet Proxy,* indem sie den Client, der im Browser ausgeführt wird, mit der iSeries verbindet. Außerhalb dieser Anwendung kann auf die 400 mit

Internet-Clients nicht zugegriffen werden. Diese Lösung ist etwas sicherer als die letzte Methode, da nur die 5250-Konnektivität, die vom Browser bereitgestellt wird, auf die iSeries Zugriff erhält. Diese Option erfordert jedoch einen zusätzlichen Dienst auf dem Server, von dem die Kommunikation zwischen Webserver und iSeries abgewickelt wird. Wie die Option für die Direktverbindung erlaubt Ihnen auch diese Lösung, die Software zentral zu verwalten. Aktualisierungen werden nur auf dem Webserver gespeichert und auf den Clients bei der Anmeldung bereitgestellt.

Beide Lösungen basieren auf Telnet 5250-Clients, die in einem Webbrowser ausgeführt werden. Der dritte Typ weist einige Unterschiede zu den anderen Lösungen auf. Dieser Typ Web-to-Host-Produkt konvertiert den 5250-Datenstrom dynamisch in eine HTML-Seite. Als Resultat erhalten Ihre iSeries-Bildschirme das Look and Feel einer Standardwebseite. Abbildung 9.3 zeigt das Ablaufdiagramm dieses Verbindungstyps.

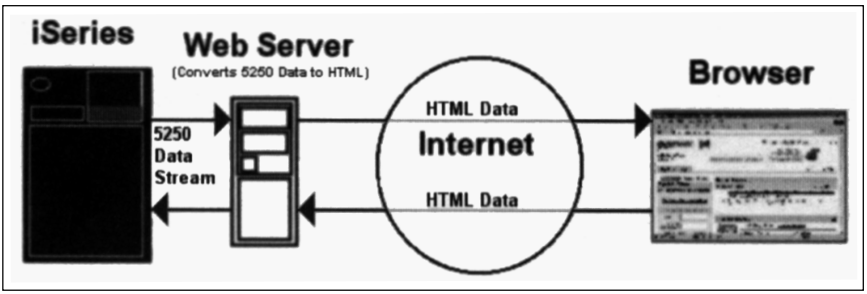


Abbildung 9.3: Dieses Diagramm zeigt eine Verbindung, von der eine 5250-zu-HTML-Konvertierung verwendet wird.

Das IBM Produkt Workstation Gateway bietet eine ähnliche Funktionalität wie ein Produkt für die HTML-Transformierung. Der wichtigste Unterschied ist, dass der Workstation Gateway direkt auf der 400 ausgeführt wird, wohingegen die meisten anderen Produkte in dieser Kategorie die Konvertierung auf einem PC-Server ausführen. Für diese Methode ist ein Java Applet oder ActiveX Control nicht erforderlich; stattdessen wird der 5250-Datenstrom in HTML übersetzt. Dies ist teilweise aufgrund der Ähnlichkeiten zwischen den beiden Datenströmen möglich. Beide Ströme senden einen Bildschirm mit Daten zum Client und senden die eingegebenen Daten zurück zum Server.

Die Antwort auf die Frage, welchen Typ Softwarepaket Sie wählen sollten, hängt davon ab, welches Ziel Sie haben. Die Anzahl der aktuell verfügbaren Web-to-Host-Pakete ist sehr groß. Tabelle 9.1 enthält eine Liste von Produkten und deren Hersteller. Gleichgültig, welches Produkt Sie wählen, Sie unternehmen einen großen Schritt, wenn Sie Ihre iSeries oder AS/400 für das Internet bereit machen.

Tabelle 9.1: Web-to-Host Software und Anbieter-Informationen

Produktnamen	Typ	Anbieter	Webseiten-URL
Webulator/400 Translator	HTML	I/Net, Inc.	www.inetmi.com
Web-ifier verbindung	Direkt-	Anota	www.anota.com
RUMBA Management Server	Verbindung via Server	NetManage	www.netmanage.com
HostFront	Verbindung via Server	Farabi Technologies	www.farabi.com
PowerTerm WebConnect	Verbindung via Server	Ericom	www.ericom.com
Host On-Demand	Direktverbindung	IBM	www.iseries.ibm.com

iSeries Web-Client

Nein, hier geht es nicht um die Ausführung von Internet Explorer oder Netscape Navigator auf der iSeries, sondern darum, die Möglichkeit zu erhalten, Informationen vom World Wide Web in Ihre Anwendungen auf der 400 zu integrieren. Die Menge von Informationen, die derzeit im Internet verfügbar sind, ist riesig. Wenn Sie die 400 mit der Funktionalität ergänzen, die erforderlich ist, um diese Informationen von einer Green-Screen-Anwendung aus zu nutzen, erweitern Sie die Einsatzmöglichkeiten Ihrer Anwendungen und gehen mit der Zeit Ihrer Anwender sparsamer um. Hierzu aktivieren Sie die Uniform Resource Locator- (URL-) Ausführung in PC5250. Dies ist ein Feature der *Hotspot*-Funktionalität, die eingesetzt werden kann, um Bereiche des PC5250-Fensters mausfähig zu machen. Das Hotspot-Setup-Fenster wird über das Menü View aufgerufen. Sie sehen diesen Bildschirm in Abbildung 9.4.

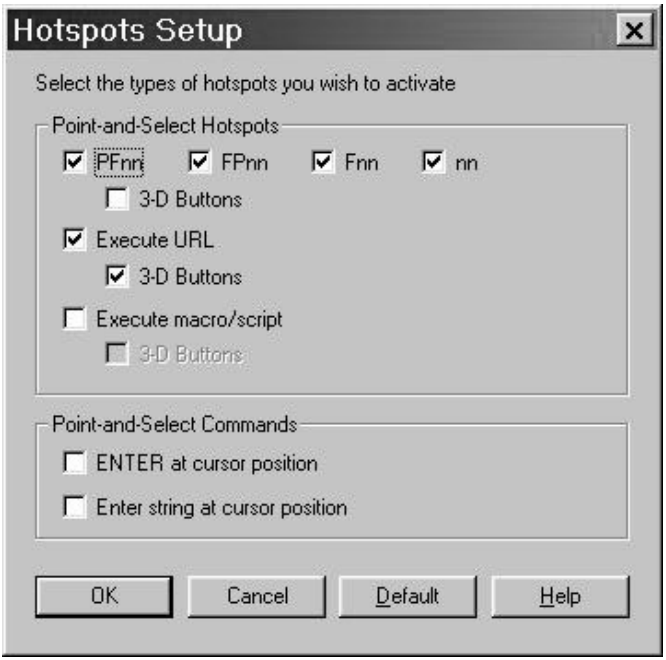


Abbildung 9.4: Dieser Bildschirm ermöglicht die Webseiten-URL-Aktivierung von PC5250.

Prüfen Sie, ob das Kontrollkästchen Execute URL aktiviert ist. Wenn Sie diese Option aktivieren, also dann wenn im PC5250-Bildschirm eine Webseiten-URL angezeigt wird (d.h.: *http://www.mcpressonline.com*), kann der Benutzer auf die URL klikken und die Webseite wird im Standard-Webbrowser des Benutzers angezeigt. Optional können Sie das Kontrollkästchen „3-D Buttons“ unterhalb von Execute URL aktivieren, um erkannte Web-URLs als Fensterschaltfläche anzuzeigen. Um diese Funktion zu testen, wechseln Sie in das PDM-Menü, indem Sie STRPDM eingeben und drücken F10, um den Bildschirm für die Befehlsein-

gabe anzuzeigen. Als Nächstes geben Sie eine Webseiten-URL ein und drücken die Eingabetaste. Es wird zwar ein Fehler angezeigt, aber ganz oben im Bildschirm noch vor der Fehleranzeige sehen Sie eine URL-Darstellung als 3-D-Schaltfläche.

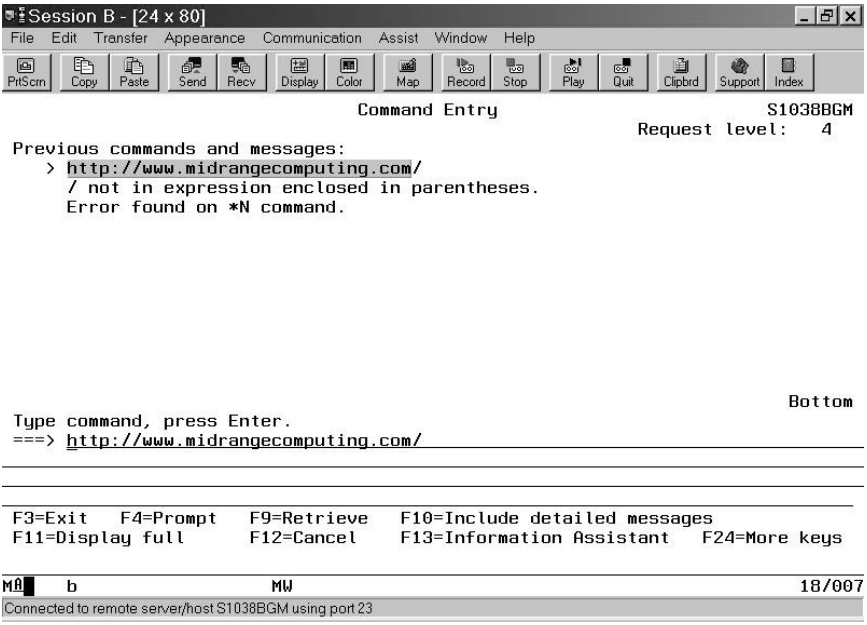


Abbildung 9.5: Webseiten-URLs können als 3-D-Schaltflächen angezeigt und mit der Maus aktiviert werden.

Auch wenn diese Tatsache nicht sofort ersichtlich ist, aber das Hinzufügen dieser Funktionalität zu einem Programm kann den Wert dieses Programms sehr erhöhen. Ein Beispiel hierfür wäre ein Auftragsanfrage-Programm, das von Ihrer Kundendienstabteilung verwendet wird. Wenn ein Kunde anruft, um herauszufinden, wo seine Werkzeuglieferung ist, muss der Kundendienstmitarbeiter in den

Auftragsanfrage-Bildschirm wechseln, die Tracking-Nummer abrufen, die Webseite des Versandunternehmens anzeigen und die Tracking-Nummer eingeben, um den aktuellen Lieferstatus abzurufen. Demgegenüber können wir diesen Prozess verkürzen, wenn wir an der Auftragsanfrage-Anwendung einige Änderungen vornehmen. Als Erstes erstellen wir eine Tabelle, die verwendet wird, um die URLs für das Versandunternehmen zu speichern. Die meisten großen Versandunternehmen geben Auskunft bezüglich der zu verwendenden URL für eine Verbindung mit ihrer Webseite. Als Nächstes ändern wir die Auftragsanfrageanwendung, so dass diese Informationen eingelesen werden und die Tracking-Nummer für den Versand an die URL angehängt wird. Abbildung 9.6 zeigt einen Codeabschnitt des ILE RPG-Codes, der hierfür verwendet wird. Diese Methode für die rasche Erstellung der Abfrage-String-Parameter weist Ähnlichkeiten mit dem in Kapitel 8 verwendeten Prozess auf.

```
 *
 *
C              CARR     Chain    CARRIERS
C                       If       %Found(CARRIERS(

C                       Eval     URL=%Trim(WEBURL) + %TRIM(TRACK)
C                       EndIf
C                       Exfmt    ORDINQ
 *
 *
```

Abbildung 9.6: Dieser Code erstellt die Webseiten-URL für die Versandverfolgung.

Wie sie sehen können, haben wir durch das Hinzufügen von weniger als zehn Zeilen Code zu dem Programm die Funktionalität sehr verbessert. Wenn nun der Kundendienstmitarbeiter den Auftragsanfrage-Bildschirm anzeigt, kann er einfach auf die URL-

Schaltfläche klicken und die Tracking-Informationen werden sofort innerhalb seines Webbrowsers angezeigt. Damit haben wir nicht nur die Effizienz der Kundendienstabteilung erhöht, sondern auch der Kunde hat nun eine bessere Meinung über das Unternehmen, weil die Reaktion auf seine Anfrage so schnell erfolgt. Ein weiteres gutes Beispiel wäre die Prüfung von Auftragsinformationen von Lieferanten.

Der große Vorteil dieser Technik ist, dass sie dazu verwendet werden kann, um neben den Webseiten auch andere Objekte anzuzeigen. Der erste Teil unserer URL (z.B. http://) wird als *Dienst* bezeichnet und dazu verwendet, den Typ „Ressource" anzugeben, mit dem wir arbeiten. Eine Liste häufig vorkommender URL-Servicetypen sehen Sie in Tabelle 9.2.

Tabelle 9.2: Uniform Resource Locator Dienste

Service	Beschreibung
http://	Hypertext Transport Protocol. Wird für Standardwebseiten verwendet.
ftp://	File Transfer Protocol. Wird für den Zugriff auf FTP-Webseiten verwendet.
telnet://	TCP/IP-Terminalemulationsprotokoll
file://	Öffnet eine angegebene Datei auf dem lokalen oder auf dem Netzwerk-Laufwerk.
nntp://	Newsgroup-Artikel-Position
gopher://	Zeigt Informationen von einem Internet GOPHER-Server an.

Weitere Informationen über URL-Dienstearten sowie allgemeine Informationen zur URL-Adressierung erhalten Sie unter *http://*

www.w3.org/Addressing/rfc1738.txt. Diese Webseite enthält Angaben des World Wide Web-Konsortiums. Diese Organisation erstellt Spezifikationen und Richtlinien in Bezug auf das Internet sowie auf internetbasierte Anwendungen. Dabei handelt es sich um eine hervorragende Ressource für alle Informationen über Internetstandards. Ein Nachteil dieses PC5250-Features ist jedoch, dass URLs, die nicht das *server://*-Format verwenden, nicht erkannt werden. URL-Server, die dieses Format nicht verwenden, sind z.B. *maito:*. Bei der Schreibweise beider Server wird Ihnen auffallen, dass die Querstriche fehlen. Für dieses Problem gibt es jedoch eine Lösung. Wenn Sie z.B. die e-Mail-Adresse eines Kunden in einen Kundenanfragebildschirm einfügen und es dem Benutzer erlauben möchten, eine e-Mail-Nachricht an den Kunden zu initiieren, indem er auf die e-Mail-Adresse klickt, müssen wir nur eine einfache Webseite erstellen, die auf die angegebene e-Mail-Adresse umgeleitet wird. Den Code für diese Seite, der mit VBScripting geschrieben wird, sehen Sie in Abbildung 9.7.

```
<%
response.write "<SCRIPT LANGUAGE=vbscript>" & CHR(10)
response.write "window.open " & CHR(34) & "mailto:" &
request.querystring("email") & CHR(34) & CHR(10)
response.write "window.close " & CHR(10)
response.write "</SCRIPT>“ & CHR(10)
%>
```

Abbildung 9.7: Dieses einfache VBScript sendet eine e-Mail an einen angegebenen Benutzer.

Dieses einfache Skript verwendet den Befehl Respons.write, um ein VBScript auf Client-Seite zu erstellen, von dem eine neue e-Mail-Nachricht angestoßen wird. Die Nachricht verwendet den Standard-e-Mail-Client für den Versand an die e-Mail-Adresse,

die in der Abfrage-String-Variable *email:* angegeben wird. Speichern Sie diese Seite auf einem ASP-kompatiblen Webserver als MAILTO.ASP. Als Nächstes verwendet die Kundenanfrageanwendung auf der 400 den in Abbildung 9.8 gezeigten Code, um den Link zur Seite MAILTO.ASP zu zeigen, die in Abbildung 9.7 erstellt wurde.

```
  *
  *
C               CUSN    Chain       CUSTMAST
C                       If          %Found(CUSTMAST)

C                       Eval        URL="http://server/mailto.asp?email='
C                                   + %TRIM(CUSEML)
C                       EndIf
C                       Exfmt       CUSINQ
  *
  *
```

Abbildung 9.8: Dieser Codeschnipsel zeigt die Seite MAILTO.ASP an.

Dieses Beispiel hängt einfach die e-Mail-Adresse des Kunden an das Ende der Webserver-URL an. Als Ergebnis wird, wenn die Anwendung angezeigt wird und ein Benutzer auf die URL klickt, der e-Mail-Client gestartet, wobei der e-Mail-Wert als TO-Adresse angezeigt wird.

Abbildung 9.9 zeigt ein Beispiel für den Kundenanfrage-Bildschirm.

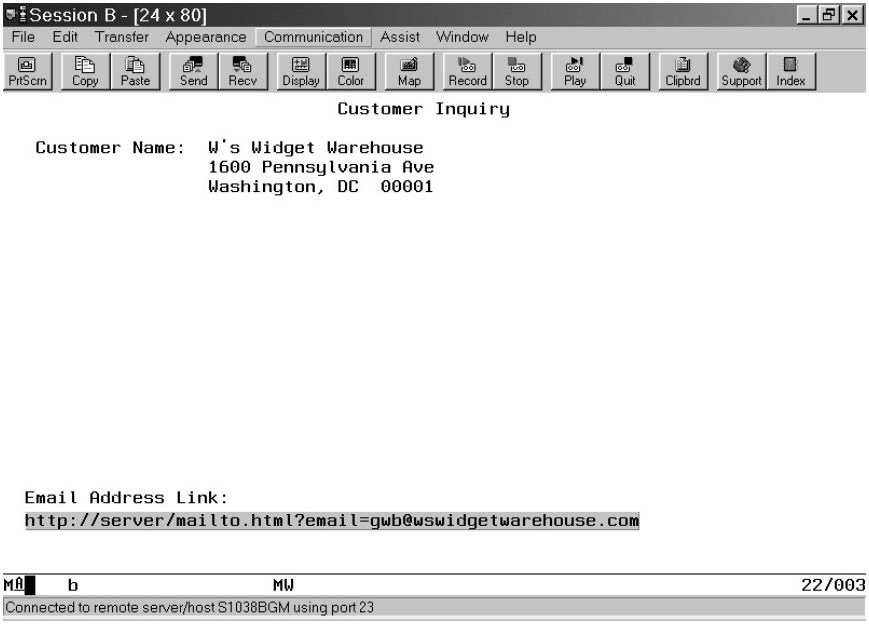


Abbildung 9.9: Der Link in diesem Bildschirm erstellt eine Nachricht für den Versand zu einer angegebenen Mail-Adresse.

Wenn der Benutzer auf die Link-Schaltfläche klickt, wird die Active Server Page MAILTO.ASP verwendet, um den Standard-Mail-Client zu öffnen, wobei die angegebene e-Mail-Adresse im Feld TO erscheint. Wie in unserem letzten Beispiel kann diese Funktion Ihnen helfen, viel Zeit zu sparen.

Einer der flexibelsten URL-Server-Präfixe ist *file://*, da wir bei der Verwendung dieses Servers dafür sorgen können, dass jeder

registrierte Dateityp auf dem Client geöffnet werden kann. Dies kann entweder ein Microsoft Excel Arbeitsblatt sein oder sogar ein Programm. Um diese Funktion zu testen, rufen Sie die OS/400-Eingabeaufforderung auf und geben Sie Folgendes ein:

```
file://C:\WINNT\NOTEPAD.EXE
```

In diesem Beispiel gehen wir davon aus, dass Ihr Client ein Windows NT-Client ist. Wenn Sie auf diese URL von Ihrer PC5250-Sitzung aus klicken, wird das Notepad-Programm gestartet.

Sehen wir uns den Prozess der Archivierung einer gespoolten Datei und der Erstellung eines Abfragesystems mit dieser Technik etwas genauer an: Als Erstes verwenden wir das ILE RPG-Programm CVTRPTRTF, das für den Befehl SNDRPTEML in Kapitel 8 erstellt wurde. Mit diesem Programm wird eine MS-Word-kompatible Version eines archivierten Berichts erstellt. Dann müssen wir einen neuen freigegebenen Ordner mit dem Namen ARCFILES erstellen. Hierzu geben wir den Befehl CRTFLR ein:

```
CRTFLR FLR(ARCFILES)
```

Die ILE RPG-Anwendung SFA001RG, die in den folgenden Illustrierungen gezeigt wird, verwendet man, um Berichte für die Archivspeicherung auszuwählen und den ausgewählten gespoolten Dateien Schlüsselwörter zuzuweisen. Diese Schlüsselwörter werden später verwendet, um nach den gewünschten gespoolten Dateien zu suchen. Die Source für die dazugehörige Anzeigedatei SFA001DS sowie für die physischen Dateien SFA001PF und SFA002PF finden Sie auf der Webseite zu diesem Buch. Diese Dateien müssen alle vor Entwicklung dieses RPG-Programms er-

stellt werden. Ebenfalls auf dieser Webseite befindet sich das CL-Programm SFA001CL, das verwendet wird, um das Programm aufzurufen. Abbildung 9.10 enthält die F- und D-Spezifikationen für das gespoolte Dateiarchivprogramm SFA001RG.

```
 *====================================================================
 * Kompilierbefehl
 *
 *       CRTBNDRPG  PGM(XXX/SFA001RG) SRCFILE(XXX/QRPGLESRC)
 *
 *====================================================================
FSFA001PF   UF A E             DISK
FSFA002PF   UF A E             DISK
FSFA001DS   CF   E             WORKSTN SFILE(SPLSFL:RR1) INFDS(SFINF)
F                                      SFILE(INDXSF:RR2)
D FILE            S              12
D ARCFLNM         S               8  0 DTAARA(SFAFILNUM)
D SFINF           DS
D KY                    369    369
D SF                    378    379B 0

D PgUp            C                   CONST(x'F4')
D PgDn            C                   CONST(x'F5')
D F3              C                   CONST(x'33')

D RR1             S               4  0
D RR2             S               4  0
D FrmtNam         S               8
D FormTyp         S              10    INZ('*ALL')
D NoRecs          S              10i 0 INZ(16)
D VAR             S            3072
D VARLEN          S              10i 0 INZ(%len(VAR))
D JbNm            S              26
D ListInf         S              80
D StRec           S              10i 0 INZ(1)
D VARSTR          S               5  0
D EOFDS           S                   LIKE(*INLR)
```

Abbildung 9.10: Die Definitionsspezifikationen für das gespoolte Dateiarchivierungsprogramm (Teil 1 von 3).

```
 *****************************************************************
 * Filter-Datenstruktur auflisten
 *****************************************************************
DFltrInf          DS          1000
D NoUserNames                    9b 0 INZ(1)
D UserID                        12
D NoOutQueues                    9b 0 INZ(1)
D OutQueue                      20
D FormType                      10
D UserData                      10
D NoStatCodes                    9b 0 INZ(1)
D StatCode                      12
D NoDevNames                     9b 0 INZ(1)
D DevName                       12
DFltrInf2         S           1000

 *****************************************************************
 * Sortire-Info Daten-Str
 *****************************************************************
DListSortInf      DS
D NoSort                 1      4B 0 INZ(0)
D SortStr                5      8B 0 INZ(0)
D SortLen                9     12B 0 INZ(0)
D DataType              13     14B 0 INZ(x'0000')
D SortOrd               15     15    INZ(x'00')
D Rsvd                  16     16    INZ(x'00')

 *****************************************************************
 *Typ-Definitionen für die Receiver Variable Formate
 *****************************************************************
DQGYV020000       DS
D QGYSFILN04             1     10
D QGYJN02               11     20
D QGYUN02               21     30
D QGYJN03               31     36
D QGYSFILN05            37     40B 0
D QGYTP00               41     44B 0
D QGYCP05               45     48B 0
D QGYCTP00              49     52B 0
D QGYOQN03              53     62
D QGYOQLN01             63     72
D QGYUSD00              73     82
D QGYTATUS01            83     92
D QGYFT01               93    102
```

Abbildung 9.10: Die Definitionsspezifikationen für das gespoolte Dateiarchivierungsprogramm (Teil 2 von 3).

```
D QGYORITY00             103    104
D QGYIJID00              105    120
D QGYISFID00             121    136
D QGYDT06                137    146
D QGYERVED36             147    160
D QGYDO                  161    167
D QGYTO                  168    173
D QGYPA                  174    174
D QGYPN                  175    184
D QGYERVED37             185    192

 ******************************************************************
 * Error Data Str
 ******************************************************************
DQUSEC DS
D QUSBPRV                  1      4B 0 inz(116)
D QUSBAVL                  5      8B 0 inz(0)
D QUSEI                    9     15
D QUSERVED                16     16
D QUSED01                 17    116

D*****************************************************************
D*Typ-Definition für Listen-Informationsformat
D*****************************************************************
DQGYOLI02          DS
D QGYTR03                  1      4B 0
D QGYRR02                  5      8B 0
D QGYRH03                  9     12
D QGYRL03                 13     16B 0
D QGYIC0                  17     17
D QGYDT07                 18     30
D QGYLS02                 31     31
D QGYRSV108               32     32
D QGYIL02                 33     36B 0
D QGYFR02                 37     40B 0
D QGYRSV207               41     80
```

Abbildung 9.10: Die Definitionsspezifikationen für das gespoolte Dateiarchivierungsprogramm (Teil 3 von 3).

Unsere Header- und Detaildateien für die Archivdaten werden zusätzlich zur Display File definiert, die von unserem Programm verwendet wird. Es werden außerdem mehrere Datenstrukturen

definiert, die von der gespoolten File List-API genutzt werden, da sie von dieser Anwendung eingesetzt wird. Sie sehen den ersten Teil der C-Spezifikationen für dieses Programm in Abbildung 9.11.

```
C     *ENTRY       PList

C                  Parm                    UsrId          10
C                  Parm                    OutQIn         10
C                  Parm                    FrmTp          10
C                  Parm                    UsrDta         10
C                  Parm                    StCd           10
C                  Parm                    DvNm           10
 *
C                  Eval      UserId=UsrId
C                  If        OutqIn'*ALL'
C     OutqIn       CAT       '*LIBL      ' OutQueue
C                  Else
C                  Eval      OutQueue=OutqIn
C                  EndIf
C                  Eval      FormType=FrmTp
C                  Eval      UserData=UsrDta
C                  Eval      StatCode=StCd
C                  Eval      DevName=DvNm

 *  List Spooled Files to Variable
C                  Movel     FltrInf       FltrInf2
C     Top          Tag
C                  Call      'QGYOLSPL'
C                  Parm                    Var
C                  Parm                    VarLen
C                  Parm                    QGYOLI02
C                  Parm                    NoRecs
C                  Parm                    ListSortInf
C                  Parm                    FltrInf
C                  Parm                    JbNm
C                  Parm      'OSPL0200'    FrmtNam
C                  Parm                    QUSEC
C                  Exsr      WriteSfl
```

Abbildung 9.11: Dieser Code zeigt eine Liste der gespoolten Dateien an, die für die Archivierungsspeicherung verfügbar sind (Teil 1 von 2).

```
C                   Dou         *INLR=*ON
C                   Eval        *IN56 = (QGYTRO3 = RR1 AND
C                                        QGYICO3 = 'C')
C                   Eval        *IN57 = (RR1=1)

C                   If          RR1=0
C                   Eval        TX='       No Records Found'
C                   Eval        Btn2='    >OK    '
C                   Eval        O1=4
C                   Eval        O2=1
C                   Eval        O3=4
C                   Exfmt       MSGBOX
C                   Eval        *INKD=*ON
C                   Else
C                   Eval        *IN30 = *on
C                   Write       PGEFT
C                   Exfmt       SPFSCT
C                   Eval        *IN31 = *off
C                   EndIf
C                   If          *INKD
C                   Eval        USRID=UserID
C                   Eval        FORMTP=FormType
C                   Eval        USRDTA=UserData
C                   Eval        STAT=StatCode
C                   Eval        PRTDEV=DevName
C                   Eval        OUTQUE=OutQIn
C                   ExFmt       SEARCH
C                   Eval        UserID=USRID
C                   Eval        FormType=FORMTP
C                   Eval        UserData=USRDTA
C                   Eval        StatCode=STAT
C                   Eval        DevName=PRTDEV
C                   Eval        OutQIn=OUTQUE
C                   If          OUTQUE'*ALL'
C     OUTQUE        CAT         '*LIBL      ' OutQueue
C                   Else
C                   Eval        OutQueue=OUTQUE
C                   EndIf
```

Abbildung 9.11: Dieser Code zeigt eine Liste der gespoolten Dateien an, die für die Archivierungsspeicherung verfügbar sind (Teil 2 von 2).

Dieser Teil des Programms verwendet die Parameter, die vom Eingabebildschirm über die *ENTRY-Parameterliste übergeben werden, um eine Liste der gespoolten Dateien anzuzeigen. Diese Dateien können anschließend für die Archivierungsspeicherung ausgewählt werden. Dies wird vom nächsten Teil des Programms übernommen, der in Abbildung 9.12 gezeigt wird.

```
 *  Liste schließen
C                   If        *INKC=*Off AND *INKL=*Off
C                   Call      'QGYCLST'
C                   Parm                    QGYRH03
C                   Parm                    QusEc
C                   Eval      *IN30=*Off
C                   Eval      RR1=0
C                   Write     SPFSCT
C                   Goto      Top
C                   EndIf
C                   EndIf
 *  F3 gedrückt -Beenden
C                   If        *INKC
C                   Eval      *INLR = *ON
C                   Else

 *  Bild-Ab gedrückt
C                   If        KY = PgDn
C                   Exsr      ReadRecs
C                   Exsr      WriteSfl
C                   Else
C                   If        KY  PgUp

 *  Verarbeiten der ausgewählten DS
C                   If        RR1>0
C                   Eval      EOFDS=*Off
C                   Dow       EOFDS=*Off
C                   ReadC     SPLSFL
C                   Eval      EOFDS=%EOF(SFA001DS)
C                   If        EOFDS=*Off
```

Abbildung 9.12: Dieser Teil des Programms steuert die Ausführung der Optionen im Archivierungsbildschirm (Teil 1 von 2).

```
C                   Select
C                   When      SEL='1'
C                   Exsr      Archive
C                   EndSl
C                   Clear                   SEL
C                   EndIf
C                   EndDo
C                   EndIf
C                   Call      'QGYCLST'
C                   Parm                    QGYRH03
C                   Parm                    QusEc
C                   Eval      *IN30=*Off
C                   Eval      RR1=0
C                   Write     SPFSCT
C                   Goto      Top
C                   Endif
C                   Endif
C                   EndIf
C                   Enddo

 *  Liste schließen
C                   Call      'QGYCLST'
C                   Parm                    QGYRH03
C                   Parm                    QusEc

C                   eval      *INlr = *on
```

Abbildung 9.12: Dieser Teil des Programms steuert die Ausführung der Optionen im Archivierungsbildschirm (Teil 2 von 2).

Dieser Abschnitt des Codes bearbeitet Befehlstastenfunktionen und listet die Subfile ein, die die gespoolten Dateien auflistet, um die Dateien zu identifizieren, die für die Archivierung ausgewählt wurden. Diese rufen die Archivierungsroutine auf.

Abbildung 9.13 enthält die Source für die vom Programm verwendeten Subroutinen.

```
 *********************************************************************
 ** Einlesen zusätzlicher Datensätze aus der Liste mit GYGTLE
 *********************************************************************
C     ReadRecs    Begsr
C                 Eval      StRec = StRec + QGYRR02
C                 Call      'QGYGTLE'
C                 Parm                    Var
C                 Parm                    VarLen
C                 Parm                    QGYRH03
C                 Parm                    QGYOLI02
C                 Parm                    NoRecs
C                 Parm                    StRec
C                 Parm                    QUSEC
C                 Endsr
 *********************************************************************
 *  Ausschreiben subfile-Datensätze für die Liste mit gespoolten Dateien
 *********************************************************************
C     WriteSfl    Begsr
C                 Eval      VarStr = 1
C                 Do        QGYRR02
C                 Eval      QGYV020000 = %subst(Var:VarStr:
C                            QGYRL03)

C                 move      QGYSFilN04    SFNAM
C                 move      QGYJN02       JOB
C                 move      QGYJN03       JOBNM
C                 move      QGYUN02       USER
C                 move      QGYTP00       PAGCNT
C                 move      QGYOQN03      OUTQ
C                 move      QGYSFILN05    FILNUM
C                 Eval      RR1 = RR1 + 1
C                 Write     SPLSFL
```

Abbildung 9.13: Subroutinen für das Einlesen der gespoolten Dateiliste, Erstellen von Subfile-Datensätzen und Archivierung ausgewählter Dateien (Teil 1 von 3).

```
 * Kein Page-Up für erste Seite
C                   Eval      *IN57 = (RR1 = 1)
C                   Eval      VarStr = VarStr + QGYRL03
C                   Enddo
C                   Endsr
 ********************************************************************
 *  Erstellen von Archivierungsdatensätzen für ausgewählte gespoolte Dateien
 ********************************************************************
C     Archive       Begsr
C                   Eval      *IN40=*Off
C                   Write     INDEX
C                   Eval      RR2=1
C                   Eval      01=1
C                   Write     INDXSF
C                   Eval      *IN40=*On
C     Top2          Tag
C                   Exfmt     INDEX
C                   If        *INKL=*Off
C                   If        *INKF=*On
C                   Eval      RR2=RR2+1
C                   Write     INDXSF
C                   Goto      Top2
C                   EndIf
C     *LOCK         In        ARCFLNM
C                   Eval      ARCFLNM=ARCFLNM+1
C                   Out(E)    ARCFLNM
C                   Move      DEL           DLTF              1
C                   Move      JOBNM         JOBNA
C                   Move      FILNUM        FNUM
C                   Eval      FILE=%CHAR(ARCFLNM) + '.RTF'
C                   Call      'SFA002CL'
C                   PARM                    JOB
C                   PARM                    SFNAM
C                   PARM                    JOBNA             6
C                   PARM                    USER
C                   PARM                    FNUM              6
C                   PARM                    FILE
C                   PARM                    DLTF
```

Abbildung 9.13: Subroutinen für das Einlesen der gespoolten Dateiliste, Erstellen von Subfile-Datensätzen und Archivierung ausgewählter Dateien (Teil 2 von 3).

```
C                   Eval      JOBNAM=JOB
C                   Eval      JOBNUM=JOBNM
C                   Eval      FILNAM=SFNAM
C                   Eval      ARCFIL='\\MYAS/400\qdls\ARCFILES\'+FILE
C                   Write     SFA001
C                   Do        RR2           X                 4 0
C     X             Chain     INDXSF                             98
C                   If        *IN98=*Off AND KEYWRD*BLANKS
C                   Write     SFA002
C                   EndIf
C                   EndDo
C                   EndIf
C                   Endsr
```

Abbildung 9.13: Subroutinen für das Einlesen der gespoolten Dateiliste, Erstellen von Subfile-Datensätzen und Archivierung ausgewählter Dateien (Teil 3 von 3).

Die erste Subroutine liest weitere Datensätze aus der geöffneten Spooldatei-Liste ein, wobei die Spooldatei-Liste API QGYGTLE verwendet wird. Die zweite Subroutine verwendet diese Daten, um Datensätze für die Subfile-Anzeige zu erstellen. Die letzte Subroutine kopiert eine ausgewählte Spooldatei in den Archivordner und erstellt Datensätze in unserer Archivdatenbank. Diese Datenbank wird später verwendet, um Datensätze zu erstellen.

In diesem Beispiel sollte der Name \\MYAS/400 mit dem Net-Server-Systemnamen für Ihr System ersetzt werden. Diesen können Sie feststellen, indem Sie die AS/400 Net-Server-Eigenschaften im Operations Navigator unter „Network/Servers/TCPIP“ anzeigen. Eine weitere Voraussetzung hierfür ist, dass wir einen Datenbereich für die Aufnahme der „nächsthöheren“ Zahl als Dateinamen für die RTF-Datei erstellen müssen.

Der Datenbereich SFLFILNUM wird erstellt, nachdem folgender Befehl Create Data Area (CRTDTAARA) eingegeben wird:

```
CRTDTAARA DTAARA(SFLFILNUM) TYPE(*DEC) LEN(8 0) VALUE(00000000)
```

Das Programm SFA001RG verwendet die API Spooled File List (QGYOLSPL), um eine Liste der gespoolten Dateien zu erstellen. Diese Liste wird mit Hilfe des Subfile-Steuerdatensatzes SPLSFL angezeigt. Den entsprechenden Bildschirm sehen Sie in Abbildung 9.14.

```
                        Archive Spooled Files

Options:1-Select for archival

Sel File Name   Job Name    Job #   File #  User ID     Pages   Out Queue
 _  SFA001DS    SFA001DS    798282  000001  MFAUST          5   QPRINTS
 _  SFA001DS    SFA001DS    798283  000001  MFAUST          5   QPRINTS
 _  FRP11U      FRP11U      799413  000001  MFAUST          3   QPRINTS
 _  QPJOBLOG    QPADEV005H  801462  000004  MFAUST         40   QEZJOBLOG
 _  SFA001DS    SFA001DS    803179  000001  MFAUST          6   QPRINTS
 _  TEST        TEST        803265  000001  MFAUST          3   QPRINTS
 _  QPJOBLOG    TEST        803265  000002  MFAUST          1   QEZJOBLOG
 _  TEST        TEST        803266  000001  MFAUST          3   QPRINTS
 _  QPJOBLOG    TEST        803266  000002  MFAUST          1   QEZJOBLOG
 _  SFA001RG    SFA001RG    803327  000001  MFAUST         20   QPRINTS
 _  QPQXPRTF    SA_NEWITMS  802787  000001  MFAUST         74   IBM6400
 _  QPQXPRTF    SA_NEWITMS  803447  000001  MFAUST         83   IBM6400
 _  OE355PT     QPADEV000T  795529  000001  MFAUST          1   IBM6400
 _  PM085PT     PM_ORDERS   798338  000001  MFAUST          1   QPRINTS
 _  PM085PT     PM_ORDERS   798342  000001  MFAUST          1   QPRINTS
 _  PM085PT     PM_ORDERS   798343  000001  MFAUST          1   QPRINTS

 F3=Exit                  Enter=Continue                  F4=Search
```

Abbildung 9.14: Gespoolte Dateien werden für die Archivierungsspeicherung in diesem Bildschirm ausgewählt.

In diesem Bildschirm werden gespoolte Dateien für die Archivierung ausgewählt, indem eine 1 neben der gewünschten Spooldatei eingegeben wird. Der Benutzer kann die Taste F4 drücken, um das Fenster für die Datensatzsuche aufzurufen (siehe Abbildung 9.15).

In diesem Fenster können die gespoolten Dateien ausgewählt werden. Die Auswahl erfolgt entweder abhängig vom Dateinamen, vom Benutzernamen, vom Jobnamen oder vom Jobdatum.

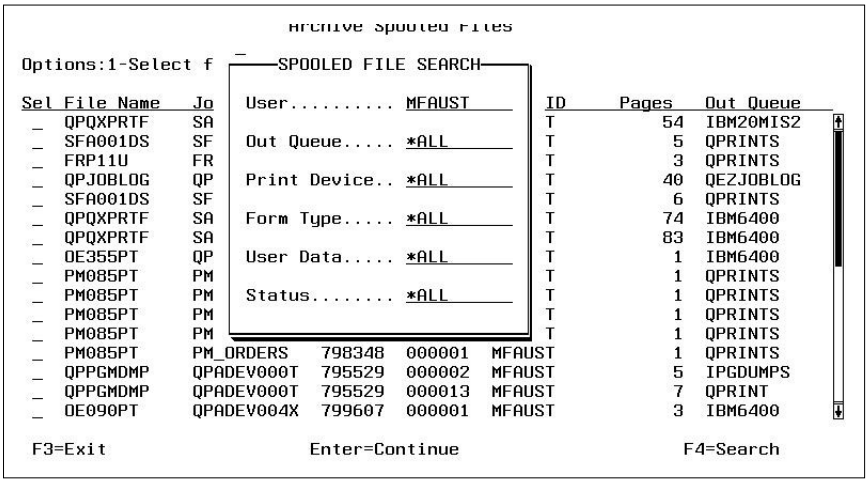


Abbildung 9.15: Die Liste der gespoolten Dateien kann in diesem Fenster gefiltert werden.

Wenn der Benutzer eine gespoolte Datei für die Archivspeicherung auswählt, wird das Fenster des Subfile-Steuerdatensatzes INDEX angezeigt. Dieses Fenster ermöglicht es dem Benutzer, bis zu 99 Schlüsselwörter für die zu indexierende Spooldatei anzugeben. Das Archiv-Schlüsselwortfenster sehen Sie in Abbildung 9.16.

Optional kann das Kontrollkästchen Delete Spooled File aktiviert werden, um die gespoolte Datei aus dem System nach der Archivierung zu entfernen. Neue Schlüsselwörter können mit der Taste F6 hinzugefügt werden. Nachdem alle erforderlichen Schlüssel-

wörter für eine gespoolte Datei eingegeben wurden, kann der Benutzer die Eingabetaste drücken, um das Objekt zu archivieren. Der Dateiname wird abhängig von einem „nächsthöheren“ numerischen Wert ermittelt, der im Datenbereich SFAFILNUM gespeichert wird.

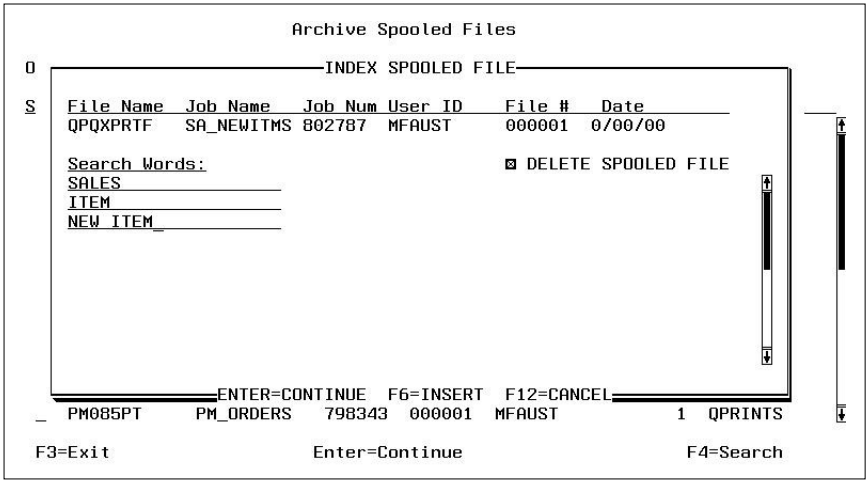


Abbildung 9.16: Dieser Bildschirm wird angezeigt, wenn eine gespoolte Datei für die Archivsteuerung ausgewählt wurde.

An diesem Punkt ruft das Programm das CL-Programm SFA001CL auf (siehe Abbildung 9.17), von dem das Programm CVTRPTRTF verwendet wird, um die gespoolte Datei in das Microsoft RTF-Format zu konvertieren. Diese Datei wird danach in den freigegebenen Ordner kopiert, der zuvor erstellt wurde. Als Nächstes erstellt das Programm einen Datensatz in der physischen Datei SFA001PF, der die Informationen zur gespoolten Datei sowie

Namen und Position der RTF-Datei enthält. Anschließend werden Datensätze für jedes der definierten Schlüsselwörter in die physische Datei SFA002PF geschrieben.

```
 /*====================================================================*/
 /* Kompilierung:                                                      */
 /*                                                                    */
 /*             CRTCLPGM   PGM(XXX/SFA002CL) SRCFILE(XXX/QCLSRC)       */
 /*                                                                    */
 /*====================================================================*/
SFA002CL:     PGM         PARM(&JOBNAM &SFNAM &JOBNUM &USER &FNUM +
                            &ARCFIL &DLTF)
              DCL         VAR(&SFNAM)  TYPE(*CHAR) LEN(10)
              DCL         VAR(&JOBNAM)  TYPE(*CHAR) LEN(10)
              DCL         VAR(&USER) TYPE(*CHAR) LEN(10)
              DCL         VAR(&JOBNUM)  TYPE(*CHAR) LEN(6)
              DCL         VAR(&ARCFIL)  TYPE(*CHAR) LEN(12)
              DCL         VAR(&DLTF)  TYPE(*CHAR) LEN(1)
              DCL         VAR(&FNUM) TYPE(*CHAR) LEN(6)

              CHKOBJ      OBJ(QTEMP/SPLFIN) OBJTYPE(*FILE)
              MONMSG      MSGID(CPF9801) EXEC(CRTPF FILE(QTEMP/SPLFIN) +
                            RCDLEN(202) SIZE(*NOMAX))
              CHKOBJ      OBJ(QTEMP/RPTOUT) OBJTYPE(*FILE)
              MONMSG      MSGID(CPF9801) EXEC(CRTPF FILE(QTEMP/RPTOUT) +
                            RCDLEN(512) SIZE(*NOMAX))

              CPYSPLF     FILE(&SFNAM) TOFILE(QTEMP/SPLFIN) +
                            JOB(&JOBNUM/&USER/&JOBNAM) SPLNBR(&FNUM) +
                            MBROPT(*REPLACE) CTLCHAR(*PRTCTL)

              CLRPFM      FILE(RPTOUT)
              CALL        PGM(CVTRPTRTF)
              CPYTOPCD    FROMFILE(QTEMP/RPTOUT) TOFLR('ARCFILES') +
                            TODOC(&ARCFIL) REPLACE(*YES)

              IF          COND(&DLTF *EQ '1') THEN(DO)
              DLTSPLF     FILE(&SFNAM) JOB(&JOBNUM/&USER/&JOBNAM) +
                            SPLNBR(&FNUM)
              ENDDO
              ENDPGM
```

Abbildung 9.17: Dieses CL-Programm erstellt eine RTF-Version einer angegebenen gespoolten Datei.

Dieses Programm verwendet die angegebenen Parameter, um die gespoolte Datei in die physische Datei SPLFIN zu kopieren, die als Eingabedatei für das Programm CVTRPTRTF dient. Die RTF-Ausgabe aus diesem Programm wird in die Datei RPTOUT eingefügt, die in den freigegebenen Ordner ARCFILES mit Hilfe des Befehls Copy to PC Document (CPYTOPCD) kopiert wurde. Wenn die CL-Variable &DEL auf 1 gesetzt wurde, wird die Spooldatei als Nächstes gelöscht. Der Befehl Archive Spooled File (ARCSPLF), den Sie in Abbildung 9.18 sehen können, wird für den Zugriff auf diese Anwendung verwendet.

```
/*  Befehl ARCHIVE SPOOLED FILE                                  */
/*                                                               */
/*  COMPILE:  CRTCMD CMD(XXX/ARCSPLF)                            */
/*            PGM(XXX/SFA001CL)                                  */
/*                                                               */

             CMD        PROMPT('Archive Spooled Files')

             PARM       KWD(USRID) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL) (*CURRENT)) MIN(0) +
                          CHOICE('Name, *CURRENT, *ALL') +
                          PROMPT('User ID')
             PARM       KWD(OUTQ) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MIN(0) CHOICE('Name, *ALL +
                          ') PROMPT(,Output Queue‘)
             PARM       KWD(FRMTP) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL) (*STD)) MIN(0) CHOICE('Form +
                          Type, *ALL, *STD') PROMPT('Form type')
             PARM       KWD(USRDTA) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MIN(0) CHOICE('User Data, +
                          *ALL') PROMPT('User Data')
             PARM       KWD(STCD) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MIN(0) CHOICE('Status +
                          code, *ALL') PROMPT('Status Code')
             PARM       KWD(DVNM) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MIN(0) CHOICE('Device +
                          Name, *ALL') PROMPT('Device name')
```

Abbildung 9.18: Dieser Befehl ruft die gespoolte Dateiarchivierungsanwendung auf.

Dieser Befehl akzeptiert vier Parameter, mit denen die gespoolten Dateien für die Anzeige ausgewählt werden, wenn die Anwendung zum ersten Mal gestartet wird. Der Befehl hat folgendes Format:

```
ARCSPLF USER(Benutzer-ID) JOBNAME(Jobname) JOBNUM(Jobnummer)
        DATE(Jobdatum)
```

Als Nächstes müssen wir die Anwendung erstellen, mit der archivierte gespoolte Dateien abgerufen werden. Das ILE RPG-Programm SFA005RG, das in den Abbildungen 9.19 und 9.20 gezeigt wird, verwendet die zuvor erstellten Daten, um es einem Benutzer zu ermöglichen, nach einer erforderlichen gespoolten Datei zu suchen. Die verknüpfte Display File SFA005DS befindet sich auf der angegebenen Webseite, gemeinsam mit der Source für die logische Datei SFA002L1, die ebenfalls von diesem Programm verwendet wird.

```
 *=============================================================
 * Kompilierbefehl
 *
 *      CRTBNDRPG  PGM(XXX/SFA005RG) SRCFILE(XXX/QRPGLESRC)
 *
 *=============================================================
FSFA001PF  IF   E           K DISK
FSFA002L1  IF   E           K DISK
FSFA005DS  CF   E             WORKSTN SFILE(SPASFL:RR1)

D EOFDS           S                   LIKE(*INLR)
D EOF01           S                   LIKE(*INLR)
D EOF02           S                   LIKE(*INLR)
D DateTemp        S               D   DATFMT(*ISO)
C     *ENTRY       PList
 * Specify Either
C                   Parm                    SpFileName        10
C                   Parm                    JobName           10
```

Abbildung 9.19: Der Kernabschnitt unserer Archivdatei-Abrufanwendung (Teil 1 von 3)

```
C                   Parm                    JobNumber          6
C                   Parm                    UserName          10
C                   Parm                    SpFileNumber       6
 *         Or
C                   Parm                    KeyWord           20
 *
C                   If        SpFileNumber='*ALL'
C                   Eval      SpFile=999999
C                   Else
C                   Move      SpFileNumber  SpFile            6 0
C                   EndIf
C                   If        JobNumber='*ALL'
C                   Eval      JobNo=999999
C                   Else
C                   Move      JobNumber     JobNo             6 0
C                   EndIf
C                   If        SpFileName*Blanks
C                   If        SpFileName='*ALL' Or JobName='*ALL'
C                   Or        JobNumber='*ALL' Or UserName='*ALL'
C                   Or        SpFileNumber='*ALL'
C                   Exsr      SelAll
C                   Else
C                   Exsr      SelByParms
C                   EndIf
C                   Else
C                   If        KeyWord*Blanks
C                   Eval      KEYWD=Keyword
C                   Exsr      ByKeyword
C                   Else
C                   Exsr      SelAll
C                   EndIf
C                   EndIf
C                   DoW       *INKC=*Off
C                   If        RR1=0
C                   Clear                   SPASFL
C                   Eval      RR1=1
C                   Write     SPASFL
C                   Eval      TX='       No Records Found'
C                   Eval      Btn2='    >OK    '
C                   Eval      O1=4
C                   Eval      O2=1
C                   Eval      O3=4
C                   Exfmt     MSGBOX
C                   EndIf
```

Abbildung 9.19: Der Kernabschnitt unserer Archivdatei-Abrufanwendung (Teil 2 von 3)

```
 *
C                   Eval      *IN30=*On
C                   Write     PGEFT
C                   Exfmt     SPASCT
 *
C                   Select
C                   When      *INKC=*On
C                   Eval      *INLR=*On
C*
C                   When      *INKE=*On
C                   If        KEYWD='*ALL' Or KEYWD=*Blanks
C                   Exsr      SelAll
C                   Else
C                   Exsr      ByKeyword
C                   EndIf
C*
C                   Other
C                   Eval      EOFDS=*Off
C                   DoW       EOFDS=*Off
C                   ReadC     SPASFL
C                   Eval      EOFDS=%EOF(SFA005DS)
C                   If        EOFDS=*Off
C                   If        SEL='5'
C     SFKEY2        Chain     SFA001PF
C                   If        %Found(SFA001PF)
C                   Eval      ARCLNK='file://' + ARCFIL
C     *ISO          Move      ARCDAT        DateTemp
C     *USA          Move      DateTemp      ARCDTE
C                   EndIf
C                   Exfmt     ARCINF
C                   Clear                   SEL
C                   Update    SPASFL
C                   EndIf
C                   EndIf
C                   EndDo
*
C                   EndSl
C                   EndDo
```

Abbildung 9.19: Der Kernabschnitt unserer Archivdatei-Abrufanwendung (Teil 3 von 3)

Dieser Abschnitt des Programms verwendet die Werte, die über die Parameterliste *ENTRY übergeben werden, um eine Liste der Datensätze aus der Archivdatenbank anzuzeigen. Das Programm zeigt eine URL für einen ausgewählten Eintrag an und ermöglicht dem Benutzer, dieses Dokument mit einem Doppelklick aufzurufen. Abbildung 9.20 zeigt die Subroutinen für diese Anwendung.

```
 ****************************************************************************
 ** Mit den *ALL Werten einlesen
 ****************************************************************************
C      SelAll       Begsr
C                   Eval      *IN30=*Off
C                   Eval      RR1=0
C                   Write     SPASCT
 *
C      *LOVAL       Setll     SFA001PF
 *
C                   Eval      EOF01=*Off
C                   DoW       EOF01=*Off
C                   Read      SFA001PF
C                   Eval      EOF01=%EOF(SFA001PF)
C                   If        EOF01=*Off
C                   If        (SpFileName'*ALL' And SpFileName=FILNAM)
C                             Or (JobName'*ALL' And JobName=JOBNAM)
C                             Or (JobNumber'*ALL' And JobNo=JOBNUM)
C                             Or (UserName'*ALL' AND UserName=USER)
C                             Or (SpFileNumber'*ALL' And SpFile=FILNUM)
C                             Or (SpFileName='*ALL' And JobName='*ALL'
C                             And JobNumber='*ALL' And UserName='*ALL'
C                             And SpFileNumber='*ALL')
C                   Eval      RR1=RR1+1
C                   Write     SPASFL
C                   EndIf
C                   EndIf
C                   EndDo
C                   Endsr
```

Abbildung 9.20: Diese Subroutinen wählen Archivierungsdatensätze basierend auf Auswahlkriterien aus (Teil 1 von 3).

```
 ****************************************************************************
 ** Mit angegebenen Parametern einlesen
 ****************************************************************************
C      SelByParms   Begsr
 *
C      SFKEY        Klist
C                   KFld                    SpFileName
C                   KFld                    JobName
C                   KFld                    JobNo
C                   KFld                    UserName
C                   KFld                    SpFile
C                   Eval       *IN30=*Off
C                   Write      SPASCT
 *
C      SFKEY        Setll      SFA001PF
 *
C                   Eval       RR1=0
C                   Eval       EOF01=*Off
C                   DoW        EOF01=*Off
C      SFKEY        ReadE      SFA001PF
C                   Eval       EOF01=%EOF(SFA001PF)
C                   If         EOF01=*Off
C                   Eval       RR1=RR1+1
C                   Write      SPASFL
C                   EndIf
C                   EndDo
C                   Endsr
 ****************************************************************************
 ** Mit Hilfe eines eingegebenen Schlüsselworts einlesen
 ****************************************************************************
C      ByKeyword    Begsr
 *
C      SFKEY2       KList
C                   KFld                    FILNAM
C                   KFld                    JOBNAM
C                   KFld                    JOBNUM
C                   KFld                    USER
C                   KFld                    FILNUM
 *
C      SFKWD        KList
C                   KFld                    KEYWD
```

Abbildung 9.20: Diese Subroutinen wählen Archivierungsdatensätze basierend auf Auswahlkriterien aus (Teil 2 von 3).

```
 *
C                   Eval      *IN30=*Off
C                   Eval      RR1=0
C                   Write     SPASCT
 *
C     SFKWD         Setll     SFA002L1
 *
C                   Eval      EOF02=*Off
C                   DoW       EOF02=*Off
C     SFKWD         ReadE     SFA002L1
C                   Eval      EOF02=%EOF(SFA002L1)
C                   If        EOF02=*Off
C     SFKEY2        Chain     SFA001PF
C                   If        %Found(SFA001PF)
C                   Eval      RR1=RR1+1
C                   Write     SPASFL
C                   EndIf
C                   EndIf
C                   EndDo
C                   Endsr
```

Abbildung 9.20: Diese Subroutinen wählen Archivierungsdatensätze basierend auf Auswahlkriterien aus (Teil 3 von 3).

Das Programm listet anfangs alle gespoolten Dateien auf, die den Parametern File Name, Job Name, Job Number, User und File Number entsprechen – oder dem Parameter Keyword aus der Parameterliste *ENTRY. Für diese Anzeige werden die Subfiles SPASFL und Control Record SPACTL verwendet. Der Bildschirm Archived Spooled File wird in Abbildung 9.21 gezeigt.

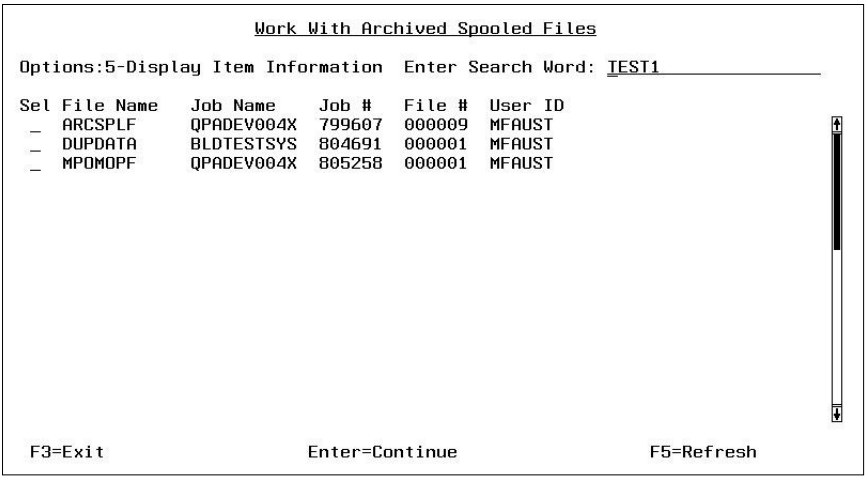


Abbildung 9.21: Diese Anzeige listet ausgewählte archivierte gespoolte Dateien auf.

Der Benutzer kann auf Grundlage eines eingegebenen Schlüsselworts eine Suche ausführen, indem er F5 drückt. Anschließend durchsucht das Programm das Archiv nach gespoolten Dateien mit passenden Schlüsselwörtern. Eine archivierte gespoolte Datei wird ausgewählt, wenn neben der gewünschten Datei eine 5 eingegeben wird. Es erscheint das Display-File-Fenster ARCINF (Abbildung 9.22).

Abbildung 9.22: Dieses Fenster enthält Informationen für eine ausgewählte gespoolte Datei sowie eine Verknüpfung mit ihrer RTF-Datei.

Wie Sie sehen, listet dieser Bildschirm die Informationen über die ausgewählte gespoolte Datei auf. Der Bildschirm verwendet auch den *file://* URL-Server, um die verknüpfte RTF-Datei zu öffnen. Diese Datei können Sie öffnen, indem Sie auf diese URL in der Anzeige klicken. Sobald die Datei in Microsoft Word angezeigt wird, können Sie sie problemlos ausdrucken.

Der Zugriff auf diese Anwendung erfolgt mit dem Befehl Work with Archived Spooled File (WRKARCSPLF). Abbildung 9.23 enthält die Source für den Befehl WRKARCSPLF.

```
/* Befehl ARCHIVE SPOOLED FILE                */
/*                                            */
/* COMPILE:  CRTCMD CMD(XXX/ARCSPLF)          */
/*                  PGM(XXX/SFA001CL)         */
/*                                            */

             CMD        PROMPT('Archive Spooled Files')

             PARM       KWD(SRCHTYPE) TYPE(*CHAR) LEN(10) +
                          SPCVAL((*SELECT) (*KEYWORD)) MIN(1) +
                          CHOICE('*SELECT, *KEYWORD') +
                          PROMPT('Search Type' 1)
             PARM       KWD(FNAME) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MIN(0) CHOICE('Name, +
                          *ALL') PMTCTL(SRCH1) PROMPT('File Name' 2)
             PARM       KWD(JOBNAME) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MIN(0) CHOICE('Name, +
                          *ALL') PMTCTL(SRCH1) PROMPT('Job Name' 3)
             PARM       KWD(JOBNUM) TYPE(*DEC) LEN(6 0) SPCVAL((*ALL +
                          0)) MIN(0) CHOICE('Number, *ALL') +
                          PMTCTL(SRCH1) PROMPT('Job Number' 4)
             PARM       KWD(USER) TYPE(*CHAR) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL) (*CURRENT)) MIN(0) +
                          CHOICE('Name, *CURRENT, *ALL') +
                          PMTCTL(SRCH1) PROMPT('User Name' 5)
```

Abbildung 9.23: Der Befehl WRKARCSPLF zeigt archivierte gespoolte Dateien an (Teil 1 von 2).

```
             PARM       KWD(FNUMBER) TYPE(*DEC) LEN(6 0) +
                          SPCVAL((*ALL 0)) MIN(0) CHOICE('Number, +
                          *ALL') PMTCTL(SRCH1) PROMPT('File Number' 6)
             PARM       KWD(KEYWORD) TYPE(*CHAR) LEN(20) MIN(0) +
                          CHOICE('Search Keyword') PMTCTL(SRCH2) +
                          PROMPT('Search Keyword' 7)
SRCH1:       PMTCTL     CTL(SRCHTYPE) COND((*EQ '*SELECT'))
SRCH2:       PMTCTL     CTL(SRCHTYPE) COND((*EQ '*KEYWORD'))
```

Abbildung 9.23: Der Befehl WRKARCSPLF zeigt archivierte gespoolte Dateien an (Teil 2 von 2).

Der erste Parameter, der mit diesem Befehl verknüpft ist, akzeptiert einen von zwei Sonderwerten, die verwendet werden, um festzustellen, welcher Typ „Auswahl" erfolgt. *SELECT wird verwendet, um gespoolte Dateien mit Hilfe einer beliebigen Kombination von Dateiname, Jobname, Jobnummer, Benutzer-ID und Dateinummer zu identifizieren. In diesem Format lautet der Befehl wie folgt:

```
WRKSPLFARC SRCHTYPE(*SELECT) FNAME(Dateinamen) JOBNAME(Jobnamen)
  JOBNUM(Jobnummer) USER(Benutzer-ID) FNUMBER(Dateinummer)
```

Der Sonderwert *KEYWORD spezifiziert ein Such-Schlüsselwort anstatt die gespoolten Datei-Jobwerte. Wenn dieser Typ der Auswahl verwendet wird, wird der Befehl – wie hier gezeigt – angegeben:

```
WRKSPLFARC SRCHTYPE(*KEYWORD) KEYWORD(search word value)
```

Bei der Anzeige des Bildschirms erscheint die verknüpfte Datei, zuerst wenn die gewünschte Datei ausgewählt und anschließend auf den Link geklickt wird. Diese Beispielanwendung ist ein nützliches Programm und ein gutes Beispiel dafür, welche Möglichkei-

ten die Anzeige von Web-URLs innerhalb eines PC5250-Bildschirms bietet. Sie können Anwendungen mit größerer Funktionalität ausstatten und gleichzeitig die Arbeit der Benutzer erleichtern.

10

Rückblick und Vorausschau

Auf den vorangegangenen Seiten wurde Ihnen eine Vielzahl von Themen vorgestellt. Im letzten Kapitel möchte ich mit Ihnen das Gelernte wiederholen und Ihnen einige zusätzliche Ideen für den Einsatz der neuen Fähigkeiten anbieten. Mit etwas Glück werden Sie noch mehr Anwendungsmöglichkeiten finden. Betrachten Sie also dieses Kapitel als krönenden Abschluss.

Kombination der Techniken

Einer der Vorteile der großen Palette von Themen, die in diesem Buch behandelt wurden, ist es, dass Sie die vorgestellten Techniken kombinieren können, um Ihre Anwendungen mit völlig neuer Funktionalität auszustatten. In Kapitel 1 wurde z.B. besprochen, wie Query Manager 400 als Ersatz für die Erstellung von RPG-Programmen für Berichte verwendet werden kann. In Kapitel 8 verwendeten wir die 400, um Dateien und Berichte via e-Mail zu senden. Ein kleiner Trick, den Sie noch nicht kennen, besteht darin, Query Manager und SMTP zu verwenden, um eine Datei via e-Mail als eine Datei im Format Comma-Separated Variable Field Length (CSV) zu versenden. Diese Datei kann von den meisten Tabellenkalkulations- oder Datenbanksoftwarepaketen gelesen werden. Es ist sogar möglich, aus einer vorhandenen Query-Management-Abfrage mit nur wenigen Änderungen ein neues Query-Manager-Report-Formular zu erstellen, mit dem eine Konvertierung in CSV möglich ist. Die Query-Management-Abfrage selbst wird modifiziert, indem ein Kommazeichen nach jedem dieser Felder eingefügt wird – mit Ausnahme des letzten Felds. Als Beispiel dient folgende SQL SELECT-Anweisung für Ihre Query-Management-Abfrage:

```
SELECT ORDNUM,BTNAME, ORDAMT FROM OELIB/ORDHDR
```

Diese Anweisung wählt Felder aus der Datei ORDHDR aus, die wir in früheren Beispielen verwendet haben. Die Ausgabe für diese Anweisung sehen Sie in Abbildung 10.1.

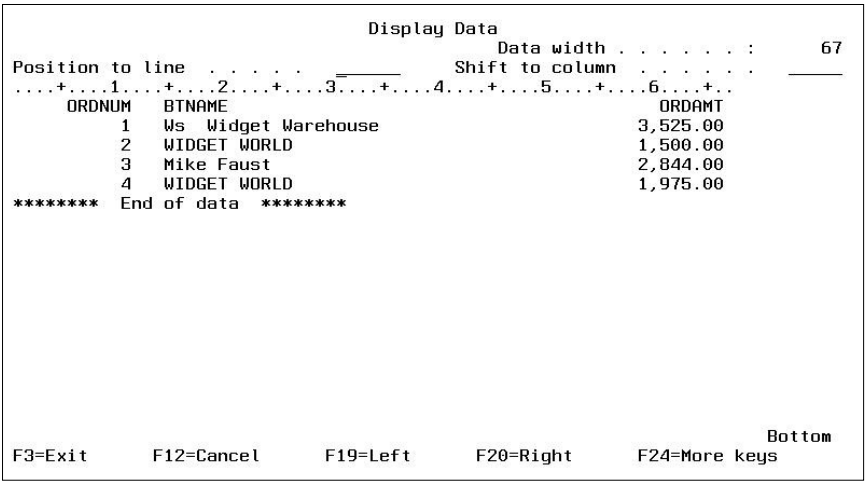


Abbildung 10.1: Unsere Beispiel-SQL-Anweisung gibt dieses Ergebnis zurück.

Um eine Konvertierung für den Export in eine CSV-Datei auszuführen, fügen wir eine Zeichenkette ein, die ein einzelnes Komma zwischen den Feldern enthält, wie Sie hier sehen können:

```
SELECT CUSTNO, ',', CUSNAM, ',' SALES FROM SALESHIST WHERE SAYEAR =
        2000
```

Das Ergebnis sieht nun etwas anders aus. Die Komma-String-Werte werden zwischen den einzelnen Feldern eingefügt, wie in Abbildung 10.2 gezeigt.

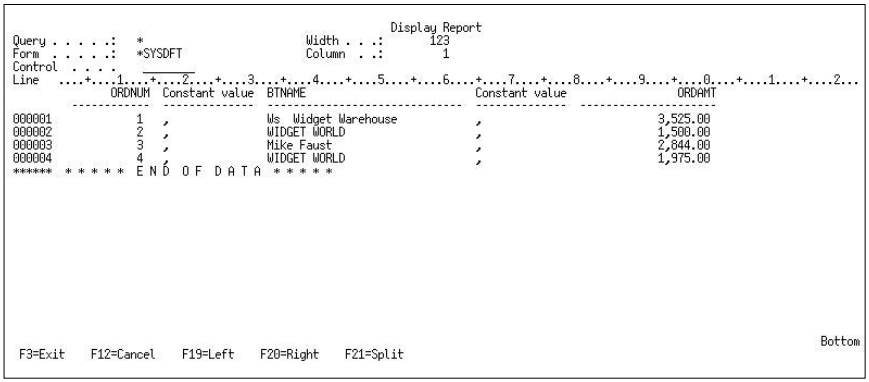


Abbildung 10.2: Die neuen String-Konstanten werden zwischen den vorhandenen Feldern eingefügt.

Als Nächstes erstellen wir ein Query-Management-Berichtsformular, das benötigt wird, um die CSV-Ausgabe zu erstellen. Wenn ein Formular bereits vorhanden ist, würden wir hierzu die Seitenüberschriften entfernen und die Spaltenüberschriften modifizieren, so dass sie in eine Zeile passen. Dann ändern wir die Feld-Edits für jede Spalte und fügen zwischen den einzelnen Spalten ein Kommazeichen ein.

Abbildung 10.3 zeigt den Bildschirm Edit Column Formatting für die Abfrage Query Management in Abbildung10.1.

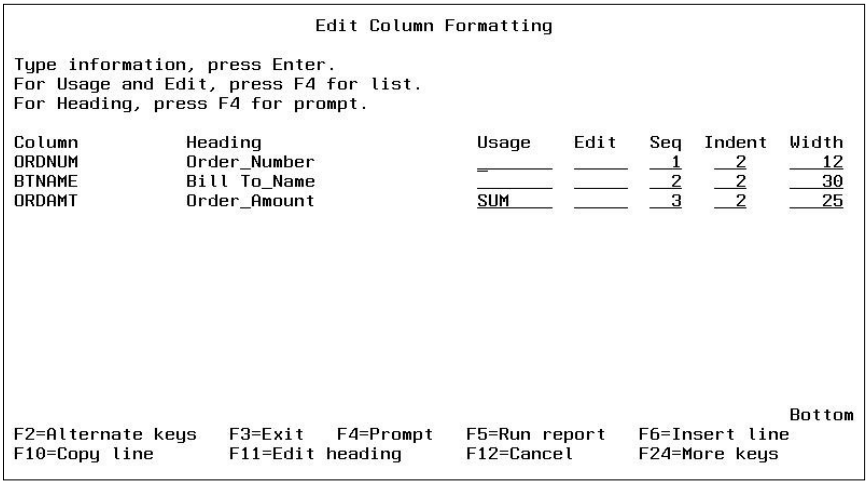


Abbildung 10.3: Der Bildschirm Edit Column Formatting für ein Formular für die Generierung eines gedruckten Berichts.

Wie Sie sehen können, wird der Unterstrich (_) angegeben, um Zeilenumbrüche zwischen den Spaltenüberschriften zu kennzeichnen. Damit sieht der gedruckte Bericht zwar besser aus, aber wenn eine CSV-Datei erstellt werden muss, gibt es Probleme, da einige Anwendungen, die CSV-Dateien – Microsoft Access zum Beispiel – einlesen, den Feldnamen von der ersten Zeile abrufen (in diesem Fall von der Spaltenüberschrift). Wenn die Spaltenüberschrift aus mehreren Zeilen besteht, werden alle Zeilen, die auf die erste Zeile folgen, fälschlicherweise als Daten interpretiert. Wir vermeiden diese Probleme, indem wir alle Unterstrichzeichen aus den

Spaltenüberschriften entfernen. Sie können den gesamten Inhalt der Überschrift so lassen, wie er ist, oder Sie können diese Überschriften kürzen. Die einzige weitere Änderung, die erforderlich ist, besteht darin, ein Kommazeichen (,) nach dem letzten Buchstaben in der Spaltenüberschrift für jede Spalte – außer für die letzte Spalte – einzufügen. Dies ermöglicht der PC-Anwendung, die Spalten auseinander zu halten, wenn die Datei eingelesen wird. Abbildung 10.4 zeigt den Edit-Column-Formatting-Bildschirm für die CSV-Version dieses Query-Management-Formulars.

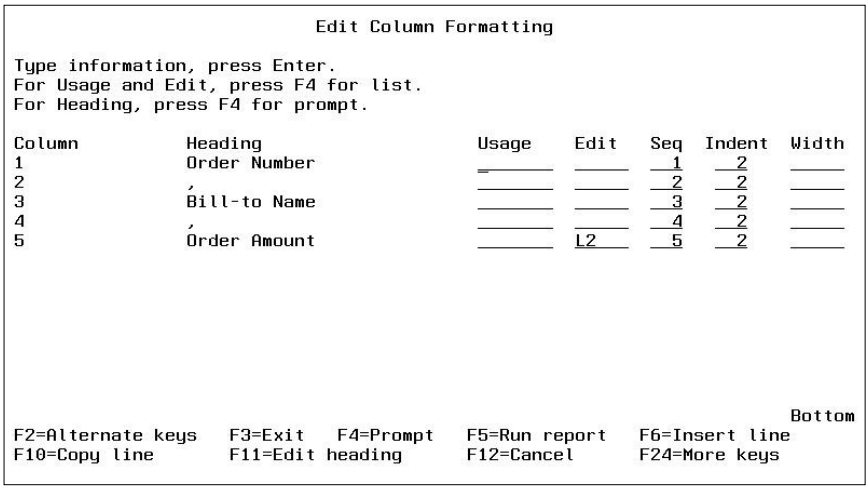


Abbildung 10.4: Dieses Formular erstellt CSV-Ausgaben aus einer Query-Management-Abfrage.

Sie werden bemerken, dass wir die Unterstriche aus den Spaltenüberschriften entfernt haben. Außerdem haben wir die Spaltenüberschrift, die unseren beiden Feldern zugewiesen wurde, einfach mit einem Komma (,) ersetzt. Dies führt dazu, dass ein Komma

innerhalb der Spaltenüberschriften sowie in den eigentlichen Daten erscheint. Zusätzlich dazu haben wir die Funktion SUM aus dem Auftragsbetrag entfernt. Dies geschieht, da die Anwendung, die die CSV-Datei einliest, sonst diese Zeile irrtümlicherweise als zusätzliche Zeile einlesen würde. Wir müssen außerdem einen Edit-Code für alle numerischen Daten verwenden, so dass keine Kommas als Tausender-Separatoren ausgedruckt werden. In diesem Fall verwenden wir L2, um festzulegen, dass zwei Dezimalstellen ausgedruckt werden sollen. Eine weitere Aufgabe, die wir erledigen müssen, ist die Anzeige des Bildschirms Specify Formatting Options in der Anzeige Select Report Format für die Anzeige von Spalten-Separatoren. Abbildung 10.5 zeigt diesen Bildschirm und die Werte, die festgelegt werden müssen.

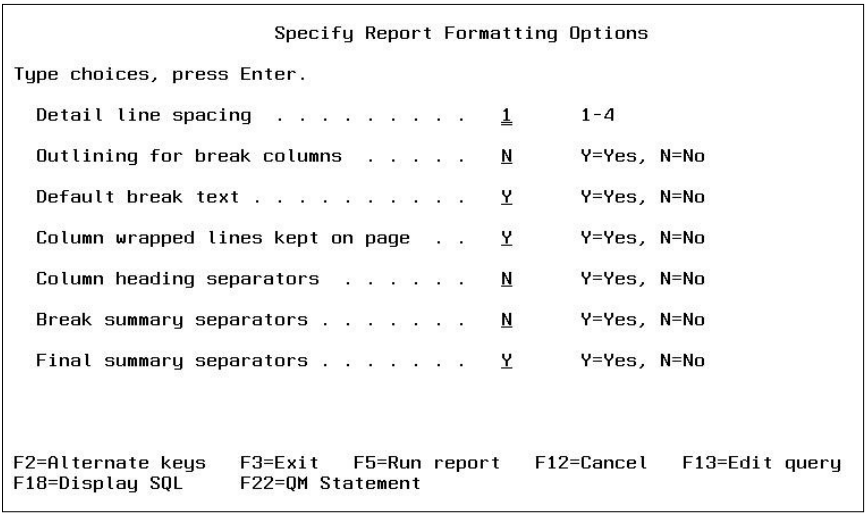


Abbildung 10.5: Die Spalten-Heading-Separatoren müssen bei der Erstellung von CSV deaktiviert werden.

Wir haben die Option Column Heading Separators auf No gesetzt, um zu verhindern, dass eine Zeile Bindestriche zwischen den Spaltenüberschriften und den Daten eingefügt. Auch das geschieht, weil die Anwendung, die die Datei einliest, diese Zeile fälschlicherweise für CSV-Daten halten könnte. Abbildung 10.6 zeigt die Ausgabe dieses Query-Management-Berichtsformulars.

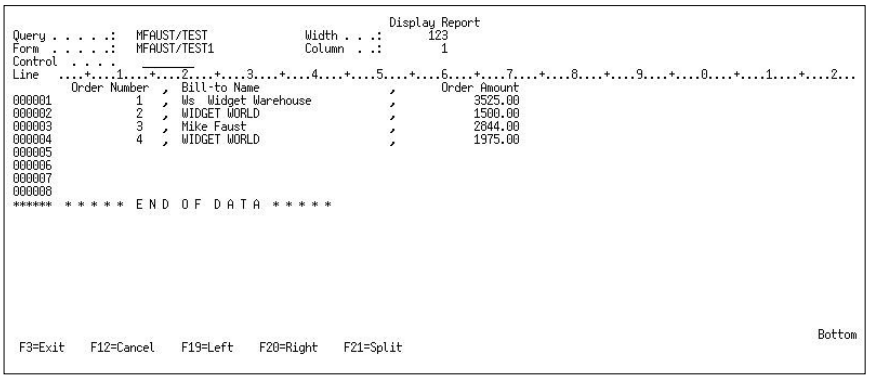


Abbildung 10.6: Das Resultat zeigt unsere Daten im CSV-Format.

Vorteilhaft ist, dass wir die modifizierte Version der Query-Management-Abfrage mit der ursprünglichen Version für die Erstellung von gedruckten Berichten verwenden können, indem wir einfach das Schlüsselwort OMIT in die Komma-Spalten eingeben.

Nachdem wir die Daten formatiert haben, müssen wir sie für den Versand vorbereiten. Dies geschieht mit dem gleichen dreistufigen Prozess, mit welchem wir den Bericht via e-Mail versendet haben:

1. Kopieren Sie die gespoolte Dateiausgabe mit CPYSPLF in eine physische Datei.

2. Kopieren Sie mit CPYTOPCD die physischen Dateidaten in ein Dokument in einen freigegebenen Ordner.
3. Versenden Sie das Dokument aus dem freigegebenen Ordner mit SNDDST.

Bevor wir Schritt 1 ausführen können, müssen wir eine Datei für die Ausgabe erstellen. Verwenden Sie hierzu den Befehl Create Physical File (CRTPF):

```
CRTPF FILE(CSVOUTPUT) RCDLEN(512)
```

Es empfiehlt sich, zu prüfen, ob die Datensatzlänge ausreicht, um die einzelnen Datensätze Ihrer Ausgabe aufzunehmen. Sie können auch die Seitenbreite für das Query-Management-Formular mit dem Befehl Override Printer File (OVRPRTF) überschreiben:

```
OVRPRTF PAGESIZE(255 378)
```

Dieses Beispiel erhöht die Seitenlänge auf maximal 255 Zeilen und die Seitenbreite auf maximal 378 Zeichen. Damit wird verhindert, dass nach 66 Zeilen umgebrochen und Spaltenüberschriften erzeugt werden. Das kann jedoch geschehen, wenn mehr als 255 Zeilen Daten innerhalb einer Abfrage generiert werden. Der Rest von Schritt 1 wird ausgeführt, indem zuerst die Abfrage Query Management ausgeführt wird und dann die gespoolte Dateiausgabe mit dem Befehl Copy Spooled File (CPYSPLF) kopiert wird:

```
CPYSPLF FILE(QPQXPRTF) TOFILE(MFAUST/CSVOUTPUT)
        JOB(805225/MFAUST/QPADEV005H)
        SPLNBR(*LAST)
```

Hiermit wird die Ausgabe des Query-Management-Formulars in die zuvor erstellte physische Datei kopiert. In Schritt 2 werden diese Daten aus der physischen Datei in ein Dokument in einem freigegebenen Ordner kopiert, indem der Befehl Copy to PC Document (CPYTOPCD) wie folgt eingegeben wird:

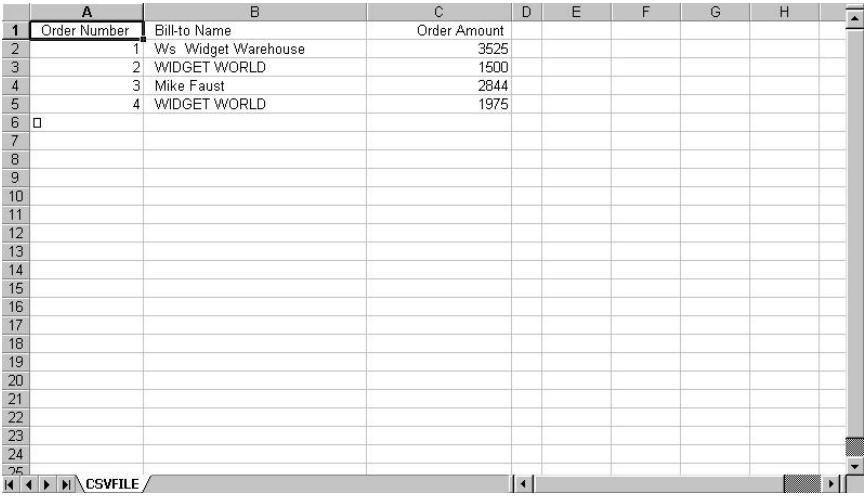
```
CPYTOPCD FROMFILE(MFAUST/CSVOUTPUT) TOFLR(TEMP) TODOC(CSVFILE.CSV)
```

Dieser Befehl fügt die physischen Dateidaten, die in Schritt 1 erstellt wurden, in das Dokument CSVFILE.CSV im freigegebenen Ordner ein. Das ist wichtig, da wir nur so die PC-Dateierweiterung .CSV mit der Datei verknüpfen können. Diese Erweiterung wird von der e-Mail-Client-Software benutzt, um festzustellen, welche Anwendung für das spätere Öffnen der Datei auf dem PC eingesetzt werden soll. Der dritte Schritt besteht darin, dieses Dokument mit dem hier gezeigten Befehl Send Distribution (SNDDST) zu versenden:

```
SNDDST TYPE(*DOC) TOINTNET(()) DSTD(,CSV File‘)
        MSG(Hier ist die CSV-Datei) DOC(CSVFILE.CSV) FLR(TEMP)
```

Wenn die Datei auf dem Client empfangen wird, kann sie entweder von einer beliebigen Anwendung geöffnet werden, die für die Handhabung dieses Dateityps konfiguriert wurde, oder sie kann für die spätere Verwendung gespeichert werden.

Abbildung 10.7 zeigt ein Beispiel dafür, wie die Daten aus unserem Beispiel innerhalb einer Tabellenkalkulationsanwendung aussehen.



	A	B	C	D	E	F	G	H
1	Order Number	Bill-to Name	Order Amount					
2	1	Ws Widget Warehouse	3525					
3	2	WIDGET WORLD	1500					
4	3	Mike Faust	2844					
5	4	WIDGET WORLD	1975					
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								

CSVFILE

Abbildung 10.7: CSV-Dateianzeige in Microsoft Excel

Wie Sie sehen, eröffnen sich damit völlig neue Verwendungsmöglichkeiten für diese beiden Techniken. Wir können eine weitere Technik hinzufügen, die wir bereits kennen gelernt haben, um diese Funktionalität stark zu erweitern. In Kapitel 2 beschäftigten wir uns mit der Frage, wie das Feld SNGCHCCTL für die Erstellung von Optionsschaltflächen verwendet werden kann. Wir könnten z.B. einen Befehlseingabebildschirm erstellen, der ein Einfachauswahlfeld enthält. Dieses ermöglicht dem Benutzer, auszuwählen, dass der Bericht an einen Drucker gesendet oder für e-Mail im CSV-Format gespeichert wird. Abbildung 10.8 enthält die Source einer Display File für diesen Zweck.

In diesem Beispiel wird ein Einfachauswahlfeld mit zwei Optionen erstellt: Drucker und e-Mail (CSV-Format). Zusätzlich zeigt die Anzeige – abhängig von Bedingungen – das e-Mail-Adressfeld an, basierend auf der Auswahl der e-Mail-Option. Abbildung 10.9 zeigt das CL-Programm mit diesem Bildschirm, der erforderlich ist, um den entsprechenden Bericht auf Grundlage der Angaben in der Anzeige auszuführen.

```
 *============================================================
 * Kompilierung
 *
 *       CRTDSPF    FILE(XXX/ORP001DS) SRCFILE(XXX/QDDSSRC)
 *
 *============================================================
A                                      DSPSIZ(24 80 *DS3)
A                                      PRINT CA03
A           R FORMAT01
A                                   1 34'Auftragsbericht'
A                                   4  4'Bericht senden an:'
A             RPTTO        2Y 0B  4 22SNGCHCFLD
A                                      CHOICE(1 'Drucker')
A                                      CHOICE(2 'E-Mail (CSV Format)')
A  79         EMLADR      38A  B  7 23CHECK(LC)
A  79                                  DSPATR(PC)
A  79 82                               ERRMSG('E-Mail-Adresse erforderlich')
A  79                               7     4'E-Mail-Adresse:'
A                                  23     4'F3=Beenden'
A                                         COLOR(BLU)
```

Abbildung 10.8: Diese Display File ermöglicht es dem Benutzer anzugeben, wohin der Bericht gesendet werden soll.

```
 /*=================================================================*/
 /* Kompilierung:                                                   */
 /*                                                                 */
 /*           CRTCLPGM   PGM(XXX/ORP001CL) SRCFILE(XXX/QCLSRC)      */
 /*                                                                 */
 /*=================================================================*/
ORP001CL:   PGM
            DCLF       FILE(ORP001DS)
TOP:        SNDRCVF    DEV(*FILE) RCDFMT(FORMAT01)
            CHGVAR     VAR(&IN82) VALUE('0')
            IF         COND(&RPTTO *EQ 2) THEN(DO)
            IF         COND(&IN79 *EQ '0') THEN(DO)
            CHGVAR     VAR(&IN79) VALUE('1')
            GOTO       CMDLBL(TOP)
            ENDDO
            IF  (&EMLADR *EQ ' ') THEN(DO)
            CHGVAR     VAR(&IN82) VALUE('1')
            GOTO       CMDLBL(TOP)
            ENDDO
            ENDDO
            IF         COND(&RPTTO *EQ 1) THEN(DO)
            STRQMQRY   QMQRY(*LIBL/ORP001QMQ) QMFORM(ORP001QMF)
            ENDDO

            IF         COND(&RPTTO *EQ 2) THEN(DO)
            STRQMQRY   QMQRY(*LIBL/ORP001QMQ) QMFORM(ORP001QCSV)
            CRTPF      FILE(QTEMP/CSVOUTPUT) RCDLEN(512)
            CPYSPLF    FILE(QPQXPRTF) TOFILE(QTEMP/CSVOUTPUT) +
                         JOB(*) SPLNBR(*LAST)
            CPYTOPCD   FROMFILE(QTEMP/CSVOUTPUT) TOFLR(TEMP) +
                         TODOC(ORDERRPT.CSV)
            SNDDST     TYPE(*DOC) TOINTNET((&EMLADR)) DSTD('Auftrags- +
                         Bericht') MSG('Angehängter Auftragsbericht') +
                         DOC(ORDERRPT.CSV) FLR(TEMP)
            DLTSPLF    FILE(QPQXPRTF) JOB(*) SPLNBR(*LAST)
            DLTF       FILE(QTEMP/CSVOUTPUT)
            ENDDO

            ENDPGM
```

Abbildung 10.9: Dieses CL-Programm erlaubt es dem Benutzer, den Bericht via e-Mail zu versenden oder einfach auszudrucken.

Dieses CL-Programm verwendet SNGCHCFLD RPTTO, um anzugeben, ob der Bericht zum Drucker oder via e-Mail versendet werden soll (CSV-Format). Das Programm verwendet Indikatoren für die Konfiguration der e-Mail-Adressfeldanzeige. Die Ausgabe dieses Bildschirms sehen Sie in Abbildung 10.10.

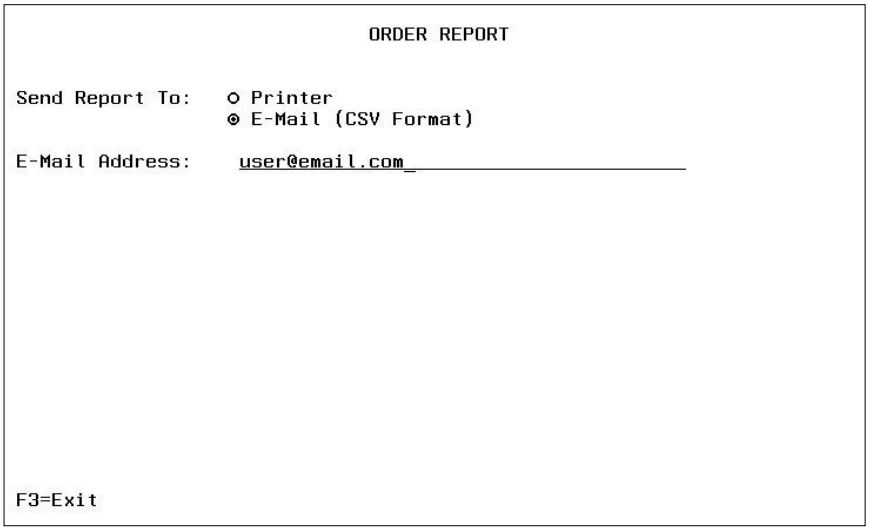


Abbildung 10.10: Dieser Bildschirm wird verwendet, um anzugeben, ob ein Bericht ausgedruckt oder via e-Mail versendet wird.

Das Programm verwendet das Ergebnis des Einfachauswahlfelds, um festzustellen, welches Query-Management-Formular verwendet werden soll. Wenn die e-Mail-Option ausgewählt ist, kopiert das Programm die erstellte Ausgabe in ein Dokument in einem freigegebenen Ordner mit einer .CSV-Erweiterung, das an die angegebene e-Mail-Adresse gesendet werden soll. Dieses Beispiel

kombiniert drei der Techniken früherer Kapitel. Die Kombination derartiger Techniken kann Ihnen dabei helfen, Ihren AS/400- und iSeries-Anwendungen ein vollständig neues Gesicht zu verleihen.

ILE-Funktion Convert Date

In Kapitel 3 wurde besprochen, wie benutzerdefinierte ILE RPG-Funktionen erstellt werden. Die Funktion Window Tools erlauben es Ihnen, Funktionen – wie z.B. einen Popup-Kalender – im Handumdrehen in Ihre Anwendungen zu integrieren. Aber nicht alle benutzerdefinierten Funktionen sind für den Endanwender sichtbar. Ein gutes Beispiel hierfür ist die Konvertierung eines Datumsformats in ein anderes Format – z.B. die Konvertierung eines intern gespeicherten Datums im Format *yyyymmdd* in ein Format für die Anzeige von *mmddyy*. Normalerweise lautet der hierfür erforderliche Code wie folgt:

```
C     *ISO      Move      FromDt    DateFld
C     *MDY      Move      DateFld   ToDate
```

In diesem Beispiel wird das numerische Datumsfeld FromDate im *ISO Format (*yyyymmdd*) in das Datumsformatfeld DateFld konvertiert. Als Nächstes konvertieren wir dieses Datum in das Format *MDY (*mmddyy*) im numerischen Datumsfeld ToDate. Obwohl diese Technik nur zwei Zeilen Code benötigt, kann sie weiter vereinfacht werden. Dies geschieht mit der Funktion CVTDATE, mit der diese Konvertierung in der folgenden einzelnen Codezeile ausgeführt werden kann:

```
C                Eval      ToDate=CvtDate(FromDate:"*YYMD": "*MDY")
```

Obwohl nur eine Zeile Code aus dem obigen Beispiel entfernt wurde, kann mit der Option, diesen Code in eine einzelne Funktion zu integrieren, noch viel mehr erreicht werden: Sie haben nun die Möglichkeit, Datumsangaben in uneinheitlichen Formaten zu verglei-

chen, ohne dafür eine der Datumsangaben in ein Feld konvertieren zu müssen. Der folgende Code z.B. kann programmiert werden, um das Feld SLSDTE, das das Format *yyyymmdd* aufweist, mit dem Feld CURDTE, das das Format *mmddyy* aufweist, zu vergleichen:

```
C                   If        CvtDate(SLSDTE: "*YYMD":"*MDY")=CURDTE
```

Die Möglichkeit, diesen Funktionstyp innerhalb eines ILE RPG-Programms zu vereinfachen, kann Ihnen viel Zeit ersparen. Die knapp gehaltene Funktion Convert Date in Abbildung 10.11 verhilft Ihnen dazu.

```
HDATFMT(*ISO)
 *===================================================================
 * Befehl kompilieren
 *
 * Befehle erstellen: CRTRPGMOD MODULE(library/CDT001 )
 *                              SRCFILE(library/QRPGLESRC)
 *
 *===================================================================
D CvtDate          PR             8  0
D  FromDate                       8  0 VALUE
D  FromFmt                        5    VALUE
D  ToFmt                          5    VALUE
P CvtDate          B
D CvtDate          PI             8  0
D  FromDate                       8  0 VALUE
D  FromFmt                        5    VALUE
D  ToFmt                          5    VALUE
D DateFld          S               D
D Err              S                    LIKE(*INLR)
 *
C                     If        FromDate0
C                     Eval      Err=*Off
```

Abbildung 10.11: Diese ILE RPG-Funktion konvertiert die verschiedenen Datumsformate (Teil 1 von 3).

```
 *
 * 'FROM'-Format verarbeiten
 *
C                   Select
C                   When      FromFmt='*YYMD'
C                   *ISO      Test(DE)      FromDate
C                   If        %Error = *Off
C     *ISO          Move      FromDate      DateFld
C                   Else
C                   Eval      Err=*On
C                   EndIf
C                   When      FromFmt='*YMD'
C     *YMD          Test(DE)                FromDate
C                   If        %Error = *Off
C     *YMD          Move      FromDate      DateFld
C                   Else
C                   Eval      Err=*On
C                   EndIf
C                   When      FromFmt='*MDY'
C     *MDY          Test(DE)                FromDate
C                   If        %Error = *Off
C     *MDY          Move      FromDate      DateFld
C                   Else
C                   Eval      Err=*On
C                   EndIf
C                   When      FromFmt='*CYMD'
C     *CYMD         Test(DE)                FromDate
C                   If        %Error = *Off
C     *CYMD         Move      FromDate      DateFld
C                   Else
C                   Eval      Err=*On
C                   EndIf
C                   When      FromFmt='*MDYY'
C     *USA          Test(DE)                FromDate
C                   If        %Error = *Off
C     *USA          Move      FromDate      DateFld
C                   Else
C                   Eval      Err=*On
C                   EndIf
C                   EndSl
```

Abbildung 10.11: Diese ILE RPG-Funktion konvertiert die verschiedenen Datumsformate (Teil 2 von 3).

```
 *
 * Process 'TO' Format
 *
C                     If        Err=*Off
C                     Select
C                     When      ToFmt  ='*YYMD'
C     *ISO            Move      DateFld       ToDate          8 0
C                     When      ToFmt  ='*YMD'
C     *YMD            Move      DateFld       ToDate
C                     When      ToFmt  ='*MDY'
C     *MDY            Move      DateFld       ToDate
C                     When      ToFmt  ='*CYMD'
C     *CYMD           Move      DateFld      ToDate
C                     When      ToFmt  ='*MDYY'
C     *USA            Move      DateFld      ToDate
C                     EndSl
C                     EndIf
C                     Eval      Err=*On
C                     EndIf
C                     If        Err=*On
C                     Eval      ToDate=0
C                     EndIf
C                     Return    ToDate
P CvtDate           E
```

Abbildung 10.11: Diese ILE RPG-Funktion konvertiert die verschiedenen Datumsformate (Teil 3 von 3).

Diese Funktion prüft als Erstes das angegebene FROM-Format, um festzustellen, ob die angegebenen Daten für dieses Format gültig sind. Als Nächstes konvertiert sie die angegebenen Daten aus diesem Format in ein Datumsformatfeld. Wenn das angegebene Datum gültig ist, konvertiert die Funktion die Daten in das angegebene TO-Format, das als Wert der Funktion zurückgegeben wird. Wenn das angegebene Datum fehlerhaft ist, gibt die Funktion nur 0 zurück. Dieses Beispiel ist zwar recht schlicht, doch es ermöglicht Ihnen, sich eine Vorstellung davon zu machen, wie unglaublich nützlich benutzerdefinierte Funktionen sein können.

DB2/400 und SQL Server

In Kapitel 5 haben wir uns mit der Frage befasst, wie ActiveX Datenobjekte und OLE DB genutzt werden können, um PC-Anwendungen zu erstellen, die Daten aus einer DB2/400-Datenbank einlesen und Daten in eine DB2/400-Datenbank schreiben können. Vernachlässigt wurde dabei die Funktionalität, die es erlaubt, Daten von dieser Datenbank auf einen SQL-Server zu laden. Dies kann eine großartige Option sein, wenn Sie Informationen von Ihrer DB2/400-Datenbank für PC-Anwendungen verfügbar machen möchten – wie z.B. Ad-hoc-Reporting-Tools oder sogar Webseiten. Gleichzeitig möchten Sie aber die Ressourcen schonen, die von OLE DB-Diensten stark beansprucht werden. Der SQL Server ermöglicht es Ihnen, einfache Bulk-Daten-Übertragungen oder kompliziertere Übertragungen mit Hilfe der Data Transformation Services (DTS) des SQL Servers auszuführen. Am besten daran ist, dass SQL Ihnen ermöglichen wird, Importfunktionen für die tägliche, wöchentliche oder beliebige Ausführung zu planen.

Der DTS Wizard ist der beste Startpunkt für die Erstellung eines neuen Importjobs. Der SQL-Server kennt diese Jobs unter der Bezeichnung DTS Packages. Diese Anwendung erlaubt es Ihnen, Datenübertragungsanforderungen an/von einer OLE DB- oder ODBC-Datenquelle zu senden. Sie können dieses Programm nicht nur verwenden, um Übertragungen an einen SQL-Server auszuführen, sondern auch für die Übertragung an jede andere kompatible Datenquelle. Der DTS Wizard wird in der SQL-Server-Programm-Gruppe im Windows-Programmmenü als Import and Export Data (Daten importieren und exportieren) angezeigt. Abbildung 10.12 zeigt die DTS Wizard-Anzeige beim Start.

Abbildung 10.12: Der DTS Wizard wird für die Erstellung von Datenübertragungspaketen verwendet.

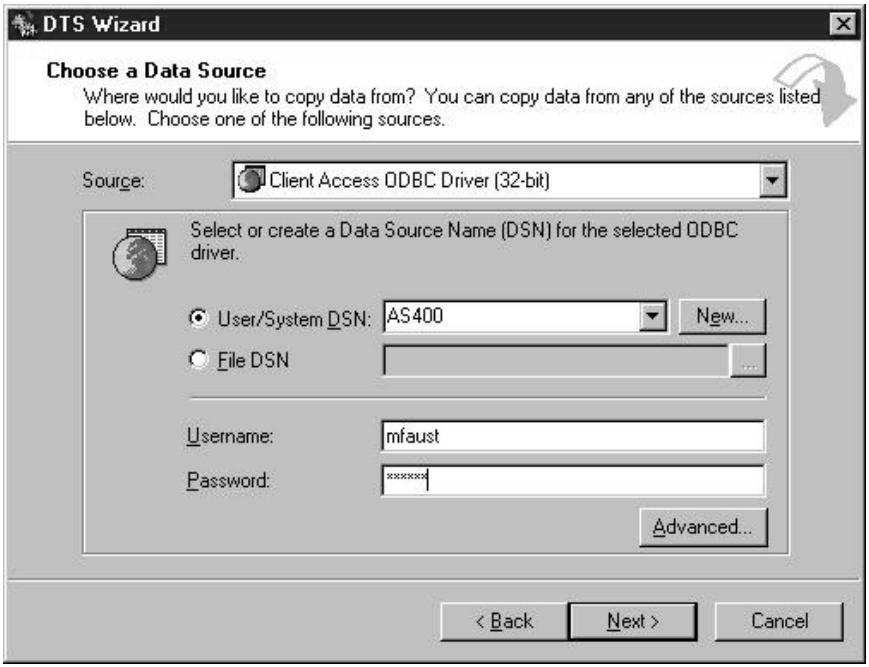


Abbildung 10.13: In diesem Bildschirm wird die Daten-Source FROM festgelegt.

Dieser Wizard kann auch vom SQL Server Enterprise Manager gestartet werden, indem der Benutzer mit der rechten Maustaste auf die gewünschte Datenbank klickt und Import Data auswählt. Nachdem er im ersten Bildschirm auf Next geklickt hat, ermöglicht es ihm die Anzeige, die Source-Datenbank zu erstellen. Das ist die Datenbank, aus der wir Daten kopieren. Im Beispiel der Abbildung 10.13 wird eine Kopie aus einer Client Access ODBC-Daten-Source mit der Bezeichnung AS/400 kopiert.

Obwohl wir in diesem Beispiel eine vorhandene Client Access ODBC-Datenquelle einsetzen, kann es sich dabei ebenso gut um eine Microsoft Access-Datenquelle, einen anderen SQL-Server oder eine beliebige ODBC- oder OLE DB-kompatible Datenquelle handeln. Wenn Sie auf die Schaltfläche Next klicken, wird der dritte Bildschirm des DTS Wizard angezeigt, den Sie in Abbildung 10.14 sehen.

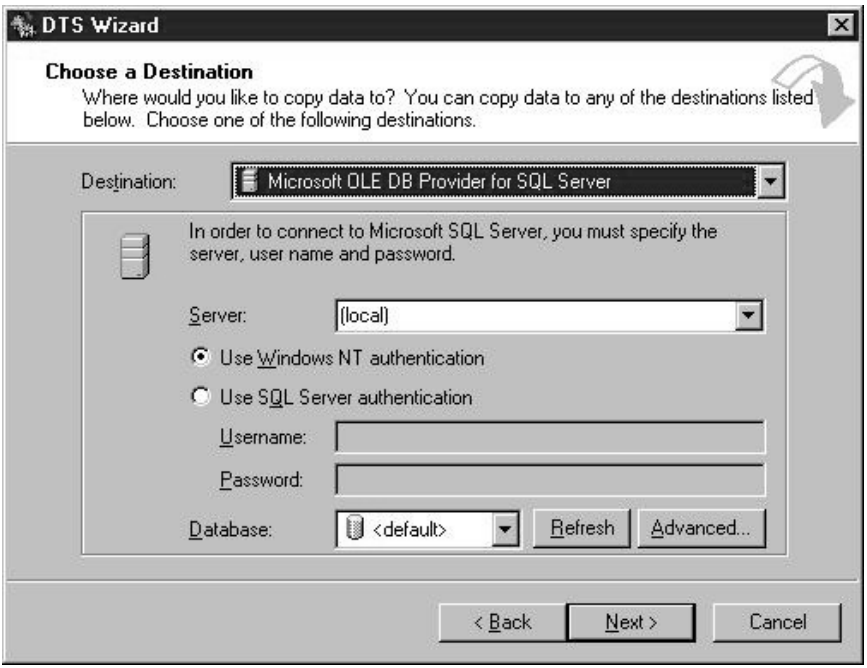


Abbildung 10.14: In diesem Bildschirm legen Sie den Zielort für die Daten-Source fest.

Die Datenquelle TO wird etwa auf die gleiche Weise definiert wie die FROM-Datenquelle. Wählen Sie einfach die entsprechende OLE

DB- oder ODBC-Datenquelle aus dem Listenfeld. In unserem Beispiel aktivieren wir die Option [local] für die Angabe, dass die Daten auf dem lokalen SQL-Server gespeichert werden sollen. Sie möchten eventuell auch die Zieldatenbank in der Dropdown-Liste mit der Bezeichnung Database ändern. Optional können Sie aus dieser Liste [New] auswählen, um eine neue Datenbank zu erstellen, die die importierten Daten enthält. Nachdem Sie Ihre Wahl getroffen haben, klicken Sie auf Next. Abbildung 10.15 zeigt den nächsten Bildschirm. Sie können in diesem Bildschirm auswählen, wie Sie die zu importierenden Daten definieren möchten.

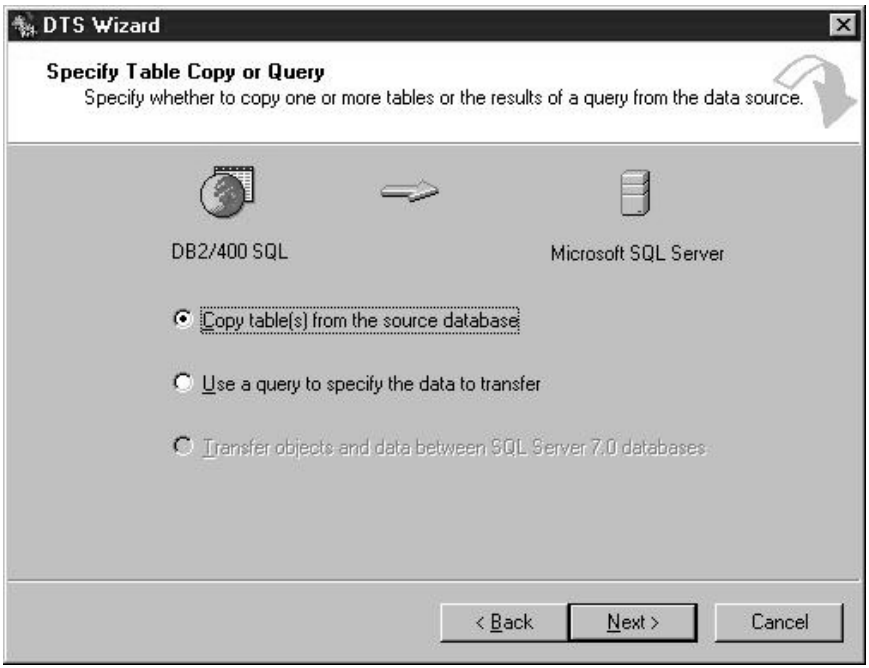


Abbildung 10.15: Wählen Sie hier, wie die zu importierenden Daten definiert werden sollen.

Wenn Sie hier die erste Option auswählen, können Sie einfach Tabellen und Felder aus einer Explorer-Liste auswählen. Die zweite Option kann aktiviert werden, um die zu kopierenden Daten mit Hilfe einer Abfrage zu definieren, die mit der einfachen SQL-Anweisung SELECT erstellt wird. Die dritte Option wird für Objektübertragungen zwischen zwei SQL-Server-Datenbanken verwendet. Klicken Sie auf die zweite Optionsschaltfläche und dann auf Next, um den Bildschirm anzuzeigen, der es Ihnen ermöglicht, eine SQL-Anweisung zu definieren. Dieser Bildschirm wird in Abbildung 10.16 gezeigt.

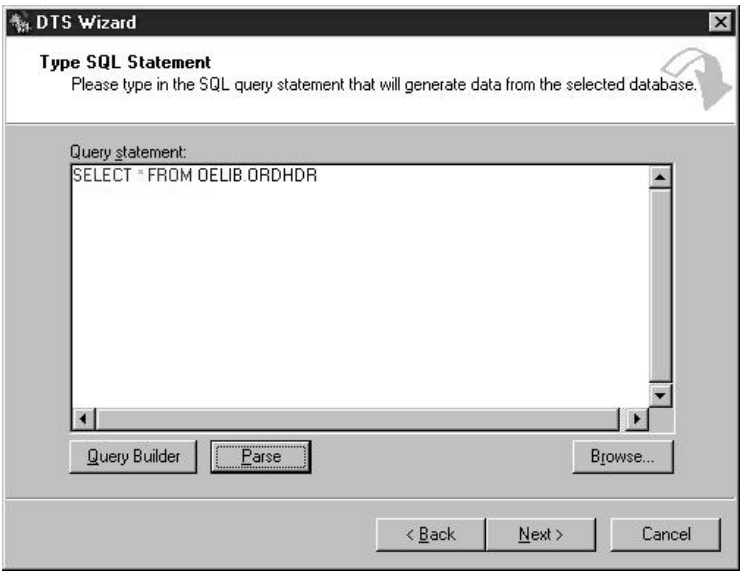


Abbildung 10.16: Die Daten unserer Source-Datenbank können als SQL-Statement definiert werden.

In diesem Beispiel wählen wir alle Datensätze aus der Datei ORDHDR in Bibliothek OELIB aus. Nachdem Sie die gewünsch-

te Anweisung eingegeben haben, klicken Sie auf die Schaltfläche Parse, um die Syntax der SQL-Anweisung zu prüfen. Sie können auch den Query Builder verwenden, um die Anweisung in einer grafischen Query Builder-Schnittstelle festzulegen. Nachdem Sie das Query Statement definiert haben, klicken Sie auf Next.

Das nächste Fenster definiert, wie die Daten von der Datenbank auf die Zieldatenbank abgebildet werden. Dieser Bildschirm kann genutzt werden, um die Daten in ein neues Format zu konvertieren und um festzulegen, welche Source-Felder in die einzelnen Zielfelder eingefügt werden sollen. Abbildung 10.17 zeigt diese Anzeige.

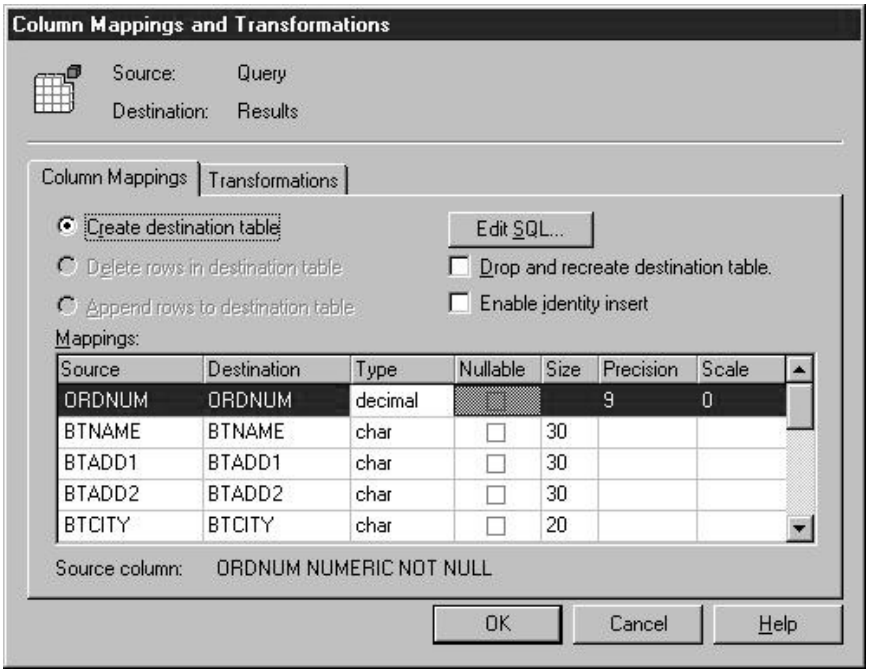


Abbildung 10.17: Die Source-Daten können mit diesem Bildschirm auf die Zieldatenbank übertragen werden.

Wenn die Tabelle für die Übertragung in der Ziel-Daten-Source nicht vorhanden ist, ist die Tabellenoption Create Destination die einzige Option, die für die Auswahl verfügbar ist. Wenn die Zieltabelle vorhanden ist, sind auch die Optionen „Delete rows in destination table“ („Zeilen in Zieltabelle löschen“) und „Append rows to destination table („Zeilen an Zieltabelle anhängen“) verfügbar. Die Registerkarte Transformations, die Sie in Abbildung 10.18 sehen, ermöglicht Ihnen, festzulegen, wie die Daten mit Hilfe eines Visual Basic Scripts übertragen werden.

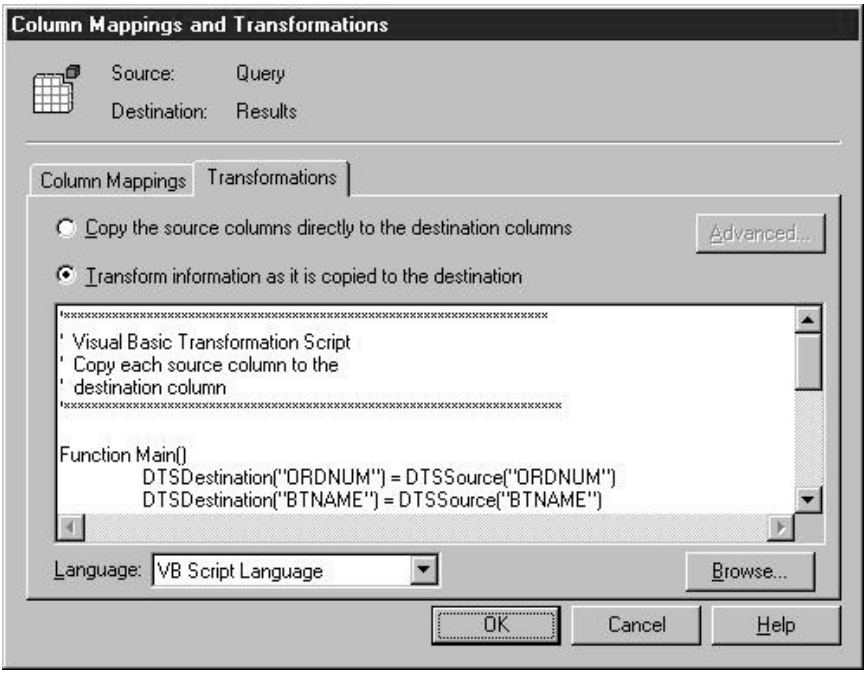


Abbildung 10.18: In diesem Bildschirm definieren Sie die Datenübertragung mit Hilfe eines VBScripts.

Optional können Sie eine JavaScript- oder XML-Transformation anstatt von VBScript ausführen. Klicken Sie auf OK und klicken Sie dann auf Next. Der angezeigte Bildschirm wird verwendet, um festzulegen, wann die Datenübertragung ausgeführt werden soll. Wie Sie in Abbildung 10.19 sehen, haben wir die Möglichkeit, die Übertragung sofort auszuführen (Run immeditately) – oder zu einem bestimmten Zeitpunkt (Schedule...). Somit können wir ein DTS-Paket erstellen, das nachts ausgeführt wird, um Daten von der 400 herunterzuladen.

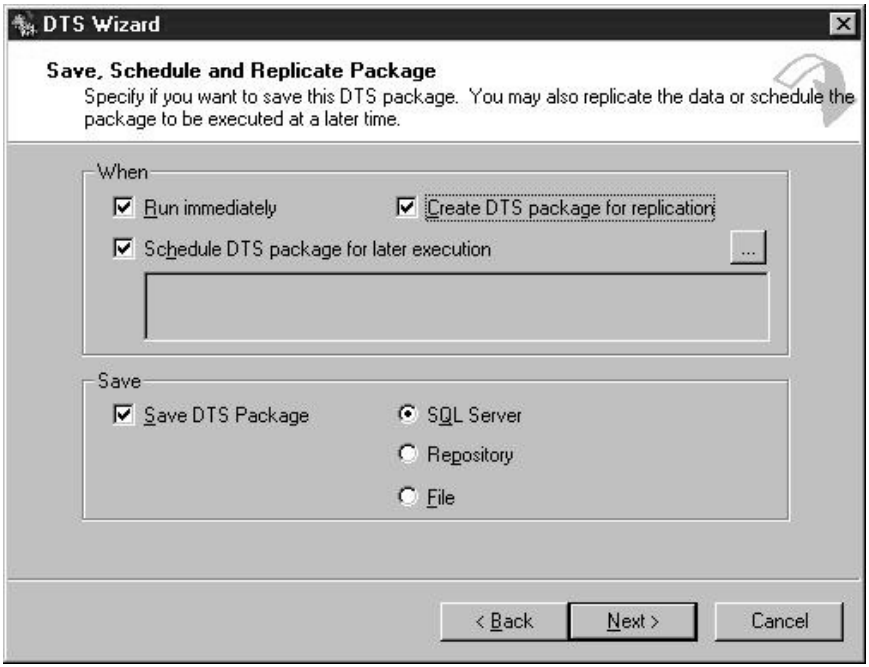


Abbildung 10.19: Ein DTS-Paket kann gespeichert und sofort oder zu einem bestimmten Zeitpunkt ausgeführt werden.

Öffnen Sie die Registerkarte Save DTS Package, um die Übertragungsdefinition entweder auf den SQL-Server, auf ein Repository oder in eine angegebene Datei zu speichern. Das Create-DTS-Paket für die Replikationsoption kann ausgewählt werden, um dieses Paket als Grundlage für eine Datenreplizierungsdefinition zu verwenden. Mit dieser können Daten zwischen zwei Systemen synchronisiert werden. Klicken Sie auf die Erstellungsschaltfläche, um den in Abbildung 10.20 angezeigten Bildschirm anzuzeigen und um einen Plan für dieses DTS-Paket zu erstellen.

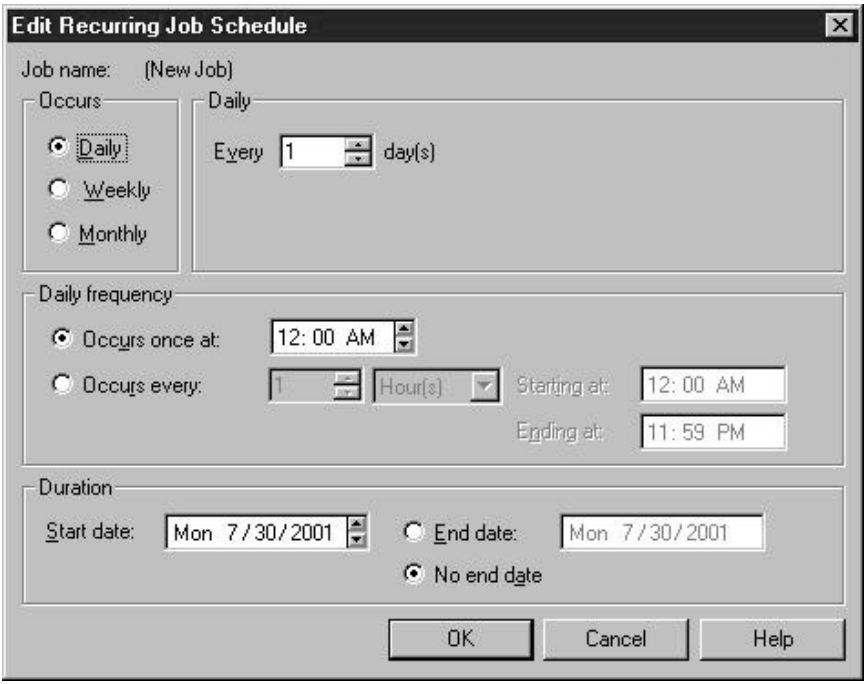


Abbildung 10.20: In diesem Bildschirm wird der Zeitpunkt für die Übertragung unseres DTS-Pakets festgelegt.

Die Planungsoptionen erlauben es Ihnen, festzulegen, wie oft die Übertragung ausgeführt wird (täglich, wöchentlich oder monatlich) und wann die Übertragung erfolgen soll – entweder in Form eines festgelegten Zeitpunkts oder eines angegebenen Inkrements. Wir können auch festlegen, zu welchem Datum dieser Job beginnen soll, und – optional – das Datum für die Beendigung der Jobausführung. Hier erhalten wir mit diesem einfachen Assistenten das erforderliche Werkzeug, um in einem angegebenen Intervall eine iSeries-Datenbank mit einem SQL-Server replizieren zu können. Sie können das DTS-Paket auch für die spätere Verwendung oder Modifizierung speichern. Abbildung 10.21 zeigt den Dialog Save DTS Package.

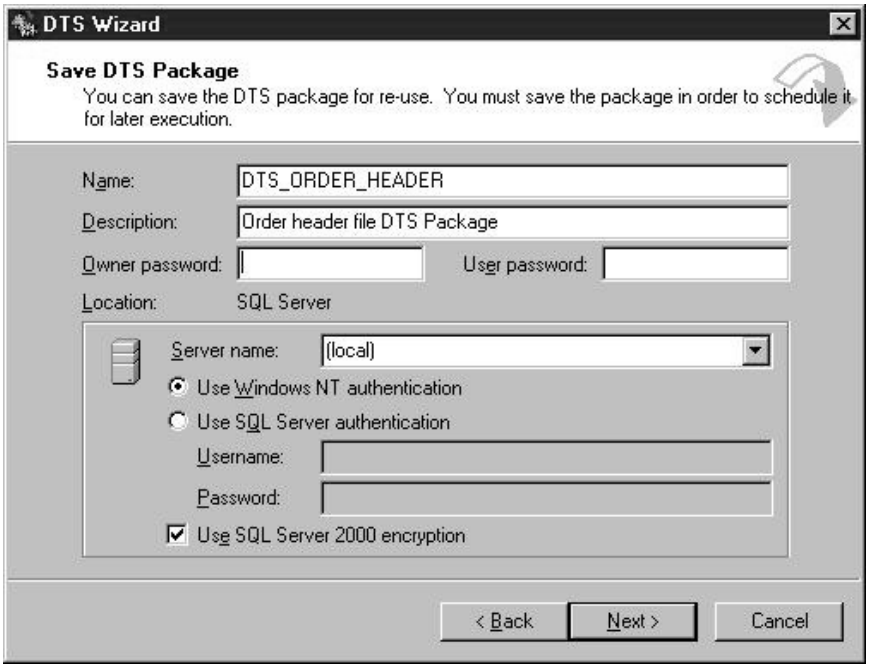


Abbildung 10.21: Das DTS-Paket kann bis zu seiner späteren Verwendung auf dem SQL Server gespeichert werden.

Das optionale Passwort ganz oben im Bildschirm kann verwendet werden, um das DTS-Paket vor unautorisierter Verwendung zu schützen. Die Informationen ganz unten im Bildschirm werden verwendet, um den SQL-Server zu definieren, auf dem das Paket gespeichert werden soll. Klicken Sie im DTS-Package-Übersichtsbildschirm auf Next und anschließend auf Finish, um die Datenübertragung zu beginnen. Der in Abbildung 10.22 angezeigte Bildschirm informiert über den Fortschritt der Ausführung des DTS-Pakets.

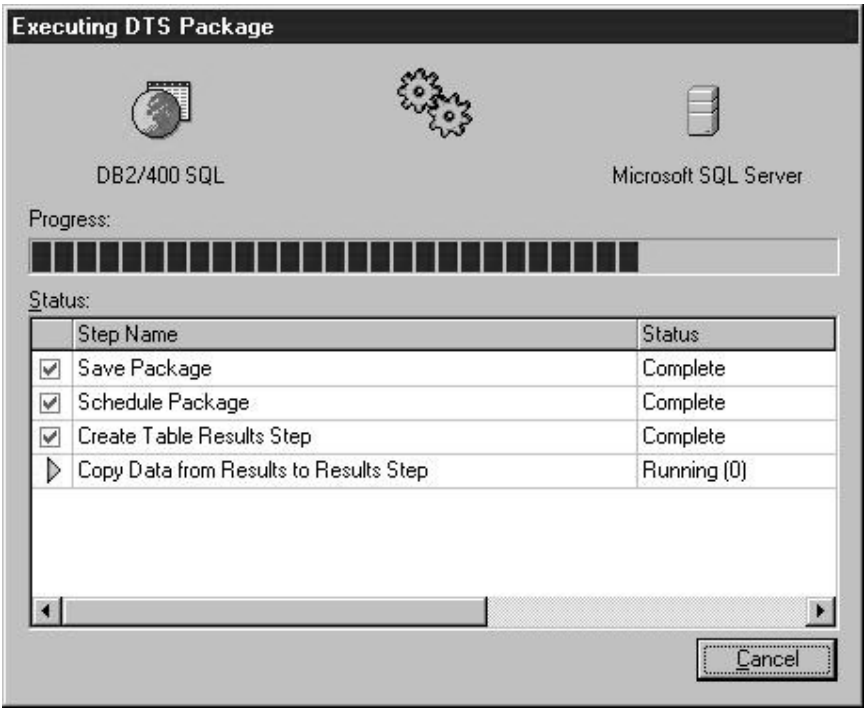


Abbildung 10.22: Dieser Bildschirm zeigt uns den Fortschritt der Ausführung des DTS-Pakets.

Es wird Ihnen sicher auffallen, dass in diesem Bildschirm jeder Teil des Prozesses ein separater Schritt ist. Die Fortschrittsleiste ganz oben im Bildschirm zeigt den Gesamtfortschritt für das gesamte Paket. Nun haben Sie eine umfassende Vorstellung davon, wie Sie mit Hilfe der Data Transformation Services eine einfache Datenübertragung von der iSeries an einen SQL-Server ausführen können.

Die Zeit vergeht im Nu

In diesem Buch wurde eine umfangreiche Palette von Techniken für die iSeries besprochen, die Ihnen sicherlich zumindest teilweise noch nicht bekannt war. Die Tatsache, dass IBM OS/400 und die iSeries weiterhin fördert, hat einen großen Teil dazu beigetragen. Wir hoffen, dass die Techniken, die Sie hier gelernt haben, Ihnen dabei helfen werden, Ihren Job zu vereinfachen und Ihre Produktivität sowie die Ihrer Anwender zu erhöhen. Ich hoffe auch, dass Ihnen das Lesen dieses Buchs ebenso viel Freude bereitet hat wie mir, als ich es schrieb.